ISSN 2311–8563

Федеральное государственное бюджетное
образовательное учреждение
высшего профессионального образования
«Самарский государственный университет»

Эвристические алгоритмы и распределённые вычисления

(Heuristic Algorithms and Distributed Computing)

Научный журнал

Том 2, выпуск 4

2015

Периодический всероссийский электронный научный журнал.

Основан в 2014 г. Выходит 4–6 раз в год.

Учредитель журнала – Самарский государственный университет.

Главный редактор – д-р физ.-мат. наук, профессор Б. Ф. Мельников.

Издательство «Самарский университет».

443011, г. Самара, ул. Академика Павлова, 1.

Тел. (846) 334-54-23.

e-mail: `university-press@samsu.ru`

Адрес редакции: 445027, г. Тольятти, ул. Юбилейная, 31-Г.

Тел. (848) 250-52-33.

e-mails: `earv@samsu.ru`, `earv-samara@yandex.ru`

Адрес страницы журнала на сайте Самарского государственного университета: `http://www.archive.samsu.ru/ru/node/4736` либо `http://samsu.ru/ru/node/4736`.

Журнал оформляется в виде pdf-файла таким образом, что его можно скачивать с сайта и распечатывать. [1]

Начиная с 2015 г., мы включаем в журнал новый раздел – «Философские проблемы математики и теоретической информатики». Сохраняются также все 4 раздела предыдущего тома: «Математическое моделирование», «Алгоритмы и эвристики», «Прикладная дискретная ма-

[1] При этом можно применить параметр «двусторонняя печать» (или аналогичный в вашей системе), после чего распечатать обложку с сайта и сброшюровать журнал обычным образом.

тематика и теория автоматов» и «Параллельные и распределённые вычисления». Особо отметим, что, как и ранее, все статьи журнала прямо или косвенно связаны с основными темами журнала – эвристическими алгоритмами и распределёнными вычислениями.

Начиная с 2015 г. вместе с журналом «Эвристические алгоритмы и распределённые вычисления» издаётся англоязычный журнал “Applied discrete mathematics and heuristic algorithms”. Однако он не является переводной версией русского издания: мы предполагаем включать в англоязычный журнал переводы некоторых русских статей за предыдущий год, однако, в первую очередь, в нём будут печататься переводы переработанных и улучшенных версий ранее напечатанных статей, а также новые оригинальные статьи.

В октябре 2015 г. Самарский государственный университет объединён с Самарским государственным аэрокосмического университета им. академика С. П. Королёва. Однако до конца 2015 г. журнал продолжит выходить в имеющемся формате.

Редакция приглашает читателей присылать статьи для обоих журналов, а также отзывы на опубликованные статьи.

Редакционная коллегия

СОДЕРЖАНИЕ

CONTENT

УДК 62–50; 519.7; 519.8

ИНТЕРВАЛЬНО-ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ*

В. И. Левин, email: vilevin@mail.ru

Пензенский государственный технологический университет

Аннотация. Теория и разнообразные методы решения дифференциальных уравнений разработаны весьма подробно. При этом всегда предполагается, что все фигурирующие в уравнениях функции являются полностью определёнными. Однако на практике задачи, сводящиеся к решению дифференциальных уравнений, часто связаны с исследованием неполностью определённых систем. Поэтому в них могут возникать дифференциальные уравнения иного типа. Фигурирующие в них искомые функции являются неполностью определёнными. Соответственно этому и производные искомых функций в дифференциальных уравнениях данного типа оказываются неполностью определёнными. Это приводит к появлению новых классов дифференциальных уравнений, в которых фигурируют неполностью определённые функции, которые требуется отыскать, и неполностью определенные производные этих функций.

Наиболее важными из этих классов дифференциальных уравнений являются вероятностные дифференциальные уравнения, в которых искомая и её производные являются случайными функциями, нечёткие дифференциальные уравнения, в которых искомая функция и её производные – нечёткие функции, и интервальные дифференциальные уравнения, где искомая функция и ее производные – интервальные функции. Первый класс дифференциальных уравнений связан с теорией вероятностей, второй класс связан с теорией нечётких множеств, а третий – с интервальной математикой, дополненной понятиями предела интервальной функции и интервальной производной.

В статье рассматривается обобщение обыкновенных дифференциальных уравнений на интервальный случай. Дается общее понятие интервально-дифференциального уравнения, его порядка, а также решения интервально-дифференциальных уравнений. Описан разработанный подход к решению интервально-дифференциальных уравнений. В частности, доказывается, что любое интервально-дифференциальное уравнение сводится к системе из двух обычных алгебраических уравнений. Дан алгоритм решения уравнений, состоящий из 5 шагов.

Ключевые слова и фразы: интервально-дифференциальное уравнение; интервальная производная; система; неопределённость; обобщение.

1 ВВЕДЕНИЕ

В различных областях науки и техники очень часто встречаются задачи, для решения которых нужно решить одно уравнение или систему

уравнений, содержащих производные искомых функций. Такие уравнения называются дифференциальными уравнениями. Теория и разнообразные методы решения таких уравнений разработаны весьма подробно [1]. При этом всегда предполагается, что все фигурирующие в уравнениях функции являются полностью определёнными. Однако на практике задачи, сводящиеся к решению дифференциальных уравнений, часто связаны с исследованием неполностью определённых систем. Поэтому в них могут возникать дифференциальные уравнения иного типа. Фигурирующие в них искомые функции являются неполностью определёнными. Соответственно этому и производные искомых функций в дифференциальных уравнениях указанного типа оказываются неполностью определёнными. Настоящая работа является введением в изучение дифференциальных уравнений именно этого типа.

2 МАТЕМАТИЧЕСКИЕ ОСНОВЫ

Будем использовать в качестве математического аппарата алгебру интервальных чисел [2], интервальный анализ [3] и интервально-дифференциальное исчисление [4].

В алгебре интервальных чисел различные операции совершаются над замкнутыми интервалами вещественных чисел, обычно определяемыми в виде множеств

$$\tilde{a} \equiv [a_1, a_2] \equiv \{a \mid a_1 \le a \le a_2\} \tag{1}$$

и рассматриваемыми как интервальные числа. Эти операции $\circ$ определяются как теоретико-множественные обобщения соответствующих операций над вещественными числами

$$\tilde{a} \circ \tilde{b} = \{a \circ b \mid a \in \tilde{a}, b \in \tilde{b}\}. \tag{2}$$

Таким образом, алгебраические операции над интервалами – сложение, вычитание, умножение, деление – вводятся в виде

$$\begin{gathered} \tilde{a} + \tilde{b} = \{a + b \mid a \in \tilde{a}, b \in \tilde{b}\},\ \tilde{a} - \tilde{b} = \{a - b \mid a \in \tilde{a}, b \in \tilde{b}\}, \\ k \cdot \tilde{a} = \{k \cdot a \mid a \in \tilde{a}\},\ \tilde{a} \cdot \tilde{b} = \{a \cdot b \mid a \in \tilde{a}, b \in \tilde{b}\},\ \tilde{a} / \tilde{b} = \{a / b \mid a \in \tilde{a}, b \in \tilde{b}\}. \end{gathered} \tag{3}$$

Из (3) вытекают формулы для вычисления результатов алгебраических операций над интервальными числами

$$\begin{gathered} \tilde{a} + \tilde{b} \equiv [a_1, a_2] + [b_1, b_2] = [a_1 + b_1, a_2 + b_2], \\ \tilde{a} - \tilde{b} \equiv [a_1, a_2] - [b_1, b_2] = [a_1 - b_2, a_2 - b_1], \\ k \cdot \tilde{a} \equiv k \cdot [a_1, a_2] = \begin{cases} [ka_1, ka_2], & k > 0, \\ [ka_2, ka_1], & k < 0, \end{cases} \\ \tilde{a} \cdot \tilde{b} \equiv [a_1, a_2] \cdot [b_1, b_2] = [\min_{i,j}(a_i \cdot b_j), \max_{i,j}(a_i \cdot b_j)], \\ \tilde{a} / \tilde{b} \equiv [a_1, a_2] / [b_1, b_2] = [a_1, a_2] \cdot [1/b_2, 1/b_1]. \end{gathered} \tag{4}$$

Объектом изучения интервального анализа являются интерваль-

ные функции [3]. Интервальная функция – однозначное отображение множества замкнутых вещественных интервалов $\{\tilde{x}\}$ (1), т.е. $\tilde{x}=[x_1,x_2]$, на аналогичное множество замкнутых вещественных интервалов $\{\tilde{y}\}$ того же вида, т.е. $\tilde{y}=[y_1,y_2]$. Символически интервальная функция записывается следующим образом:

$$\tilde{y}=\tilde{f}(\tilde{x}),\ \text{где}\ \tilde{x}=[x_1,x_2],\tilde{y}=[y_1,y_2],\ \tilde{f}(\tilde{x})=[f_1(\tilde{x}),f_2(\tilde{x})],\qquad(5)$$

где $\tilde{x}$ называется интервальной независимой переменной (интервальным аргументом), $\tilde{y}$ – интервальной зависимой переменной, $\tilde{f}$ – интервальной функцией, $f_1(\cdot)$ – нижней граничной функцией функции $\tilde{f}$, а $f_2(\cdot)$ – верхней граничной функцией интервальной функции $\tilde{f}$.

Базовым понятием интервального анализа является понятие предела интервальной функции, которое вводится следующим образом. Независимая интервальная переменная $\tilde{x}=[x_1,x_2]$ интервальной функции (5) по определению неограниченно приближается к интервалу $\tilde{x}_0=[x_{01},x_{02}]$, если при этом изменении x_1 неограниченно приближается к x_{01}, а x_2 – к x_{02}, или символически

$$(\tilde{x}\to\tilde{x}_0)\equiv(x_1\to x_{01},x_2\to x_{02}).\qquad(6)$$

Аналогично определяется неограниченное приближение зависимой интервальной переменной $\tilde{y}=[y_1,y_2]$ функции (5) к интервалу $\tilde{y}_0=[y_{01},y_{02}]$:

$$(\tilde{y}\to\tilde{y}_0)\equiv(y_1\to y_{01},y_2\to y_{02}).\qquad(7)$$

При этом если независимая переменная $\tilde{x}$ своим неограниченным приближением к интервалу $\tilde{x}_0$ вызывает неограниченное приближение зависимой переменной $\tilde{y}$ к интервалу $\tilde{y}_0$, мы говорим, что предел интервальной функции (5) при $\tilde{x}$, стремящемся к $\tilde{x}_0$, равен $\tilde{y}_0$, или в символической записи

$$\lim_{\tilde{x}\to\tilde{x}_0}\tilde{y}=\tilde{y}_0\quad\text{или}\quad\lim_{\tilde{x}\to\tilde{x}_0}\tilde{f}(\tilde{x})=\tilde{y}_0.\qquad(8)$$

Если функция (5) непрерывна, т.е. нижняя y_1 и верхняя y_2 границы зависимой переменной $\tilde{y}$ суть непрерывные функции соответственно нижней x_1 и верхней x_2 границ независимой переменной $\tilde{x}$, то предел функции (5) равен значению функции в предельной точке $\tilde{x}_0$ аргумента $\tilde{x}$, или символически

$$\lim_{\tilde{x}\to\tilde{x}_0}\tilde{f}(\tilde{x})=\tilde{f}(\tilde{x}_0).\qquad(9)$$

Основным математическим понятием, используемым в статье, является понятие интервальной производной функции [3, 4]. Оно вво-

дится на базе понятия обычной производной функции [1] следующим путём. Рассмотрим произвольную интервальную функцию $\tilde{f}$ вида (5). Будем считать её непрерывной. Зафиксируем в ней значение независимой переменной $\tilde{x}=\tilde{x}_0=[x_{01},x_{02}]$. Этому значению, в силу непрерывности функции, соответствует фиксированное значение функции $\tilde{y}_0=\tilde{f}(\tilde{x}_0)$. Теперь определим приращения независимой и зависимой переменных нашей функции относительно этих фиксированных значений:

$$\Delta\tilde{x}=\tilde{x}-\tilde{x}_0,\quad \Delta\tilde{y}=\tilde{y}-\tilde{y}_0=\tilde{f}(\tilde{x})-\tilde{f}(\tilde{x}_0), \tag{10}$$

после чего составим отношение второго приращения к первому:

$$\Delta\tilde{y}/\Delta\tilde{x}=(\tilde{y}-\tilde{y}_0)/(\tilde{x}-\tilde{x}_0)=(\tilde{f}(\tilde{x})-\tilde{f}(\tilde{x}_0))/(\tilde{x}-\tilde{x}_0). \tag{11}$$

Предел отношения (11) при неограниченном приближении независимой переменной $\tilde{x}$ к её фиксированному предельному значению $\tilde{x}_0$, если он существует, называется интервальной производной функцией от исходной интервальной функции вида $\tilde{f}(\tilde{x})$ (5) в точке $\tilde{x}_0$ и символически обозначается $\tilde{y}'_{\tilde{x}_0}$ или $\tilde{f}'_{\tilde{x}_0}(\tilde{x})$:

$$\tilde{y}'_{\tilde{x}_0}=\tilde{f}'_{\tilde{x}_0}(\tilde{x})=\lim_{\tilde{x}\to\tilde{x}_0}\Delta\tilde{y}/\Delta\tilde{x}, \tag{12}$$

где $\Delta\tilde{x}$ и $\Delta\tilde{y}$ определяются формулами (10).

Доказано [3, 4], что для существования в точке $\tilde{x}_0$ интервальной производной (12) от интервальной функции $\tilde{f}$ (5) необходимо и достаточно, чтобы в некоторой окрестности этой точки, включая её саму, значения независимой переменной $\tilde{x}$ функции $\tilde{f}$ были невырожденными интервалами (т.е. интервалами с несовпадающими верхней и нижней границами). Но вырожденность интервала $\tilde{x}$ возможных значений независимой переменной интервальной функции означает его превращение в обычную детерминированную величину. Таким образом, интервальная производная (12) от интервальной функции $\tilde{f}$ (5) существует в любой точке $\tilde{x}_0$, в которой функция $\tilde{f}$ является существенно интервальной по независимой переменной $\tilde{x}$.

Как и в случае обычной производной [1], понятие интервальной производной (12) может быть обобщено путём повторного выполнения операции взятия производной. Причём интервальная производная $\tilde{y}'_{\tilde{x}}$ из выражения (12) становится производной 1-го порядка, производная от $\tilde{y}'_{\tilde{x}}$ – производной 2-го порядка $\tilde{y}''_{\tilde{x}}$, производная от $\tilde{y}''_{\tilde{x}}$ – производной 3-го порядка $\tilde{y}'''_{\tilde{x}}$ и т.д. Производная любого n-го порядка $\tilde{y}^{(n)}_{\tilde{x}}$ задаётся следующим выражением

$$\tilde{y}_{\tilde{x}}^{(n)} = [y_{\tilde{x}}^{(n-1)}(\tilde{x})]', \quad n = 1,2,3,..., \tag{13}$$

где $\tilde{y}_{\tilde{x}}^{0}$ означает исходную интервальную функцию вида $\tilde{y} = \tilde{f}(\tilde{x})$, определяемую по формуле (5).

Согласно определениям (12), (13) интервальной производной любого порядка, все эти интервальные производные, как и исходная интервальная функция (5), при любом численном значении аргумента $\tilde{x}$ в виде некоторого интервала возможных значений $\tilde{x} = [x_1, x_2]$ также принимают численное значение в виде интервала возможных значений. Поэтому вычисление интервальной функции и интервальной производной от неё любого порядка состоит в вычислении нижних и верхних границ соответствующих интервалов. Вычисление интервальной функции $\tilde{f}$ выполняется по формуле (5), задающей указанную функцию в виде пары «нижняя f_1 и верхняя f_2 граничные функции». Интервальная производная любого n-го порядка $\tilde{y}_{\tilde{x}}^{(n)} = \tilde{f}^{(n)}(\tilde{x})$ вычисляется с помощью следующей формулы, выведенной в [3, 4]:

$$\tilde{y}_{\tilde{x}}^{(n)} \equiv \left[y_{1,\tilde{x}}^{(n)}, y_{2,\tilde{x}}^{(n)}\right] = \left[-\frac{2^{n-1}(y_2 - y_1)}{(x_2 - x_1)^n}, \frac{2^{n-1}(y_2 - y_1)}{(x_2 - x_1)^n}\right], \ \tilde{x} = [x_1, x_2], \ n = 1,2,3,... \tag{14}$$

или, по-другому,

$$\tilde{f}^{(n)}(\tilde{x}) \equiv \left[f_1^{(n)}(\tilde{x}), f_2^{(n)}(\tilde{x})\right] = \left[-\frac{2^{n-1}(f_2(\tilde{x}) - f_1(\tilde{x}))}{(x_2 - x_1)^n}, \frac{2^{n-1}(f_2(\tilde{x}) - f_1(\tilde{x}))}{(x_2 - x_1)^n}\right], \tilde{x} = [x_1, x_2], n = 1,2,3,..., \tag{15}$$

где $y_{1,\tilde{x}}^{(n)} = f_1^{(n)}(\cdot), y_{2,\tilde{x}}^{(n)} = f_2^{(n)}(\cdot)$ – нижняя и верхняя граничные функции интервальной производной функции n-го порядка $\tilde{y}_{\tilde{x}}^{(n)} = \tilde{f}^{(n)}(\cdot)$ от исходной функции $\tilde{y} = [y_1, y_2] = \tilde{f}(\tilde{x}) = [f_1(\tilde{x}), f_2(\tilde{x})]$, задаваемой формулой (5).

Как видно из формул (14), (15), интервальная производная любого n-го порядка, в отличие от обычной производной, выражается в явном виде через независимую переменную $\tilde{x} = [x_1, x_2]$ и зависимую переменную $\tilde{y} = [y_1, y_2], y_1 = f_1(\tilde{x}), y_2 = f_2(\tilde{x})$ исходной интервальной функции $\tilde{y} = \tilde{f}(\tilde{x}) = [f_1(\tilde{x}), f_2(\tilde{x})]$. Эта важная особенность интервальных производных функций принципиально упрощает теорию и методы решения интервально-дифференциальных уравнений.

Мы условимся в дальнейшем в обозначении интервальной производной $\tilde{y}_{\tilde{x}}^{(n)}$ любого порядка n оставлять обозначение точки $\tilde{x}$ лишь в случае, когда нас интересует значение производной именно в этой точке, и опускать его, записывая производную в виде $\tilde{y}^{(n)}$, в остальных случаях, когда нас интересуют её значения во всех точках.

3 ОСНОВНЫЕ ПОНЯТИЯ ОБ ИНТЕРВАЛЬНО-ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЯХ

Как хорошо известно из общей теории дифференциальных уравнений, предмет указанной теории – это те задачи, решение которых сводится к решению одного или нескольких уравнений, содержащих производные искомых функций. Такие уравнения называются дифференциальными. Более точно, дифференциальным уравнением называется соотношение, связывающее независимую переменную x, искомую функцию $y = f(x)$ и её производные различных порядков $y', y'', ..., y^{(n)}$. Если искомая функция есть функция одной независимой переменной, то дифференциальное уравнение называется обыкновенным. Если искомая функция является функцией двух или более независимых переменных, дифференциальное уравнение называется уравнением с частными производными. В настоящей работе рассматриваются только обыкновенные дифференциальные уравнения.

Порядок старшей производной, которая входит в дифференциальное уравнение, называется порядком этого уравнения. Так, дифференциальное уравнение n-го порядка имеет следующий общий вид:

$$F(x, y, y', y'', ..., y^{(n)}) = 0, \qquad (16)$$

где $F(\cdot)$ – некоторая функция от переменных в скобках. В частных случаях в уравнение (16) могут и не входить переменные x, y и отдельные производные от функции y порядка ниже, чем n, но это не изменит порядка данного уравнения, который равен n. Таким образом, уравнения $2x + 3y - 4y' = 0$ и $3y - 4y' = 0$ имеют порядок 1, уравнения $3y + y'' = 0$ и $2x + 3y - 4y' + y'' = 0$ – порядок, равный 2.

Любая функция $y = f(x)$, которая при подстановке в уравнение (16) обращает его в тождество, называется решением уравнения. Например, функция $y = e^x$ является решением уравнения $y - 2y + y'' = 0$, так как она при подстановке в это уравнение обращает его в тождество.

Будем теперь рассматривать задачи, решение которых сводится к решению уравнений (одного или нескольких), содержащих интервальные производные искомых интервальных функций, которые были введены в разделе 2. Такие уравнения мы будем называть интервально-дифференциальными. Более точно, интервально-дифференциальным уравнением мы будем называть соотношение, которое связывает независимую интервальную переменную $\tilde{x} = [x_1, x_2]$, интервальную искомую функцию $\tilde{y} = \tilde{f}(\tilde{x})$ (5) и её интервальные производные $\tilde{y}', \tilde{y}'', ..., \tilde{y}^{(n)}$. Если искомая интервальная функция является функцией одной независимой интервальной переменной, как в рассматриваемом случае, интервально-

дифференциальное уравнение назовём обыкновенным. В случае, если искомая интервальная функция является функцией двух или более независимых интервальных переменных, интервально-дифференциальное уравнение назовём уравнением с частными интервальными производными. В этой работе будем рассматривать только обыкновенные интервально-дифференциальные уравнения.

Аналогично случаю обычных (детерминированных) дифференциальных уравнений (16), порядок старшей производной, которая входит в любое интервально-дифференциальное уравнение, назовём порядком этого уравнения. Так, интервально-дифференциальное уравнение n-го порядка можно записать в следующем общем виде

$$\tilde{F}(\tilde{x},\tilde{y},\tilde{y}',\tilde{y}'',...,\tilde{y}^{(n)})=\tilde{a}\,, \qquad (17)$$

где $\tilde{F}(\cdot)$ – интервальная функция от переменных, находящихся в скобках, $\tilde{x}=[x_1,x_2]$ – независимая интервальная переменная, $\tilde{y}=[y_1,y_2]$ – зависимая интервальная переменная, находящаяся в функциональной зависимости $\tilde{y}=\tilde{f}(\tilde{x})$ от независимой переменной $\tilde{x}$; а $\tilde{y}',\tilde{y}'',...,\tilde{y}^{(n)}$ – интервальные производные порядка $1,2,3,\ldots,n$ от интервальной функции $\tilde{y}=\tilde{f}(\tilde{x})$; $\tilde{a}=[a_1,a_2]$ – числовой интервал.

В частных случаях (аналогично детерминированным дифференциальных уравнениям (16)) в интервально-дифференциальное уравнение (17) могут и не входить интервальные переменные $\tilde{x},\tilde{y}$ и отдельные интервальные производные от функции $\tilde{y}$ порядка ниже, чем n, но это не изменит порядка уравнения. Любая интервальная функция $\tilde{y}=\tilde{f}(\tilde{x})$, которая при подстановке в уравнение (17) обращает его в тождество, является решением этого уравнения. Наша задача заключается в *нахождении систематического метода и алгоритма* решения уравнения (17).

4 ЭЛЕМЕНТЫ ТЕОРИИ ИНТЕРВАЛЬНО-ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Будем называть интервальную функцию алгебраической, если она получена путём суперпозиции интервальных алгебраических операций сложения, вычитания, умножения и деления (3) над независимыми интервальными переменными указанной функции. Далее будем предполагать всегда, что интервальная функция $\tilde{F}$, связывающая переменные в интервально-дифференциальном уравнении (17), является алгебраической. В этих условиях справедлива следующая основная теорема.

Теорема 1. Любое интервально-дифференциальное уравнение ви-

да (17) эквивалентно некоторой системе из двух детерминированных алгебраических уравнений следующего вида:

$$\left.\begin{aligned} F_1(x_1, x_2, y_1, y_2) = a_1 \\ F_2(x_1, x_2, y_1, y_2) = a_2 \end{aligned}\right\}, \tag{18}$$

где $\tilde{x} = [x_1, x_2]$ – интервальная независимая переменная искомой интервальной функции $\tilde{y} = \tilde{f}(\tilde{x})$, $\tilde{y} = [y_1, y_2]$ – интервальная зависимая переменная указанной интервальной функции, F_1, F_2 – некоторые детерминированные функции, представляющие собой нижнюю и верхнюю границы интервальной связывающей функции $\tilde{F}$ уравнения (17) (т.е. $\tilde{F} = [F_1, F_2]$).

Доказательство. Согласно предположению, интервальная функция $\tilde{F}$ от переменных $\tilde{x}, \tilde{y}, \tilde{y}', \tilde{y}'', ..., \tilde{y}^{(n)}$ в уравнении (17) является алгебраической. Следовательно, она имеет вид суперпозиции элементарных операций над указанными интервальными переменными. В свою очередь, часть указанных интервальных переменных, а именно, производные $\tilde{y}', \tilde{y}'', ..., \tilde{y}^{(n)}$, по формуле (15) представляют собой суперпозиции элементарных алгебраических операций над интервальными переменными $\tilde{x} = [x_1, x_2]$, $\tilde{y} = [y_1, y_2]$, где $y_1 = f_1(\tilde{x})$, $y_2 = f_2(\tilde{x})$. Таким образом, интервальная функция $\tilde{F}$ в уравнении (17) в целом может быть представлена в виде суперпозиции элементарных алгебраических операций только над интервальными переменными $\tilde{x} = [x_1, x_2]$ и $\tilde{y} = [y_1, y_2]$.

Учитывая, что указанными элементарными операциями являются сложение, вычитание, умножение и деление, определяемые выражениями (3), и используя формулы (4) для выполнения этих операций, мы получим выражение левой части уравнения (17) в виде интервала, нижняя и верхняя границы которого представляют собой некоторые суперпозиции границ интервальных переменных $\tilde{x} = [x_1, x_2]$, $\tilde{y} = [y_1, y_2]$. Уравнение (17), таким образом, может быть переписано в явной интервальной форме

$$[F_1(x_1, x_2, y_1, y_2),\ F_2(x_1, x_2, y_1, y_2)] = [a_1, a_2], \tag{19}$$

где F_1 и F_2 – некоторые детерминированные функции. Но два интервала равны, только если равны их одноимённые границы [2]. Поэтому из (19) получаем эквивалентное ему условие (18). Но условие (19), как следует из приведенного доказательства, эквивалентно уравнению (17). Следовательно, условие (18) тоже эквивалентно уравнению (17), что и требовалось доказать. □

Как видно из теоремы 1, для решения произвольного интервально-дифференциального уравнения (17) нужно действовать следующим образом.

Алгоритм решения интервально-дифференциального уравнения.

Шаг 1. Заменить в уравнении (17) все вхождения интервальных производных $\tilde{y}',...,\tilde{y}^{(n)}$ их выражениями (15) через искомую интервальную функцию $\tilde{y}=[y_1=f_1(\tilde{x}),\ y_2=f_2(\tilde{x})]$. В результате интервально-дифференциальное уравнение (17) преобразуется в интервально-алгебраическое уравнение с неизвестной переменной $\tilde{y}$.

Шаг 2. Использовать выражения (4) для выполнения элементарных алгебраических операций над интервалами, выразить левую часть получившегося уравнения (17) в виде интервала, нижняя и верхняя границы которого имеют вид суперпозиций (функций) нижних и верхних границ интервальных переменных $\tilde{y}=[y_1,y_2]$ и $\tilde{x}=[x_1,x_2]$.

Шаг 3. Используя результаты шага 2, представить уравнение (17) в явной интервальной форме (19).

Шаг 4. Приравняв в (19) одноимённые границы левой и правой частей, перейти к эквивалентной заданному интервально-дифференциальному уравнению (17) системе двух детерминированных алгебраических уравнений (18).

Шаг 5. Решить систему уравнений (18). Найденное решение этой системы в виде $\tilde{y}=\tilde{f}(\tilde{x})$, где $\tilde{x}=[x_1,x_2], \tilde{y}=[y_1,y_2]$ ($y_1=f_1(\tilde{x}),\ y_2=f_2(\tilde{x})$) будет искомым решением интервально-дифференциального уравнения (17).)

5 ЗАКЛЮЧЕНИЕ

В этой статье показана возможность формирования дифференциальных уравнений на базе понятия интервальной производной, т.е. производной от недетерминированной функции, в которой переменные задаются с точностью до интервалов возможных значений. Новые уравнения позволяют моделировать динамику систем с интервальной неопределённостью их функций-характеристик. Главные отличия введённой интервальной производной и основанного на ней интервально-дифференциального исчисления в том, что производная любого порядка от интервальной функции снова является интервальной функцией, причём эта производная выражается в явном виде через независимую и зависимую переменные первообразной функции. Благодаря этому свойству любое интервально-дифференциальное уравнение (17) легко сводится к эквивалентной системе двух алгебраических уравнений (18), решение которых даёт решение исходного уравнения (17).

СПИСОК ЛИТЕРАТУРЫ

1. Фихтенгольц, Г. М. (2005), *Курс дифференциального и интегрального исчисления. Т. 3*, Наука, М., 728 с.

2. Алефельд, Г., Херцбергер, Ю. (1987), *Введение в интервальные вычисления*, Мир, М., 360 с.
3. Левин, В. И. (2013), «Интервальная производная и начала недетерминистского дифференциального исчисления», *Онтология проектирования*, № 4, сс. 72–84.
4. Левин, В. И. (2014), «Интервально-дифференциальное исчисление и некоторые его применения», *Информационные технологии*, № 7, сс. 3–10.

INTERVAL-DIFFERENTIAL EQUATIONS

V. I. Levin, email: vilevin@mail.ru
Penza State Technological University

Abstract. The theory and various methods for solving differential equations are developed in details. This always assumes that all functions appearing in the equations are fully defined. However, in practice problems reducing to the solution of differential equations, are often associated with the study of certain incomplete systems. Therefore, they may have another type of differential equations. Required functions of such equations are not completely certain. Accordingly, derivatives of the required functions in differential equations of this type are not completely certain. It leads to emergence of new classes of differential equations, which involve incompletely defined functions that you want to find, and not completely certain derivatives of these functions.

The most important of these classes of differential equations are stochastic differential equations where the function required and its derivatives are random functions, fuzzy differential equations where function and derivatives are fuzzy functions, and interval differential equations, where function and its derivatives are interval functions. The first class of differential equations is connected with theory of probability, the second class of differential equations is associated with theory of fuzzy sets, and the third class touches interval mathematics and concepts of limit of interval function and interval derivative.

The generalization of ordinary differential equations on the interval case is considered. The general concepts of interval-differential equation, its order, as well as the solution of interval differential equations are given. The approach to the solution of interval differential equations is developed by author. It is proved that any interval-differential equation can be reduced to system of algebraic equations. The algorithm of 5 steps to solve equations is given.

Keywords and phrases: interval-differential equation, interval derivative, system, uncertainty, generalization.

Computing Classification System 1998: G.1.0

Mathematics Subject Classification 2010: 26A24, 65G30

REFERENCES

1. Fichtengolz, G. M. (2005), *The course of differential and integral calculations, Vol. 3 [Kurs differencialnogo i integralnogo ischisleniya, Tom. 3]*, Nauka, Moscow, 728 p.
2. Alefeld, G., Herzberger, J. (1983), *Introduction in Interval Computations*, N. Y., Academic Press, 352 p.
3. Levin, V. I. (2013), “Interval derivative and beginning of nondeterministic differential calculus”, *Ontology of design* [“Intervalnaya proizvodnaya i nachala nedeterministskogo differencialnogo ischisleniya”, *Ontologiya proektirovaniya*], No. 4, pp. 72–84.
4. Levin, V. I. (2014), “Interval-differential calculation and some its application”, *Information technologies* [“Intervalno-differencialnoe ischislenie i nekotorye ego primeneniya”, *Informacionnye tehnologii*], No. 7, pp. 3–10.

ЭВРИСТИЧЕСКИЕ АЛГОРИТМЫ И РАСПРЕДЕЛЁННЫЕ ВЫЧИСЛЕНИЯ, **2**, 4 (2015) 19–30

УДК 519.816

ВЕРОЯТНОСТНЫЙ ПОДХОД К РЕШЕНИЮ МНОГОКРИТЕРИАЛЬНЫХ ЗАДАЧ С ПОМОЩЬЮ МЕТОДА АНАЛИЗА ИЕРАРХИЙ *

М. А. Беляев, email: mbelyaev1991@gmail.com
Самарский государственный аэрокосмический университет им. академика С. П. Королёва

Аннотация. В работе рассматривается метод анализа иерархий и его проблемы выхода значений матрицы парных сравнений за фундаментальную шкалу предпочтений. Метод анализа иерархий позволяет найти лучший из альтернативных вариантов, распределить ресурсы между альтернативами пропорционально их приоритетам, а также может быть использован для сравнения сложных объектов в управлении качеством или в психолого-педагогическом тестировании. Данный метод основывается на парных сравнениях, в совокупности позволяющих ранжировать альтернативы на основе их попарного сравнения, выполняемого экспертами. Ввиду влияния различных «человеческих факторов» сравнения альтернатив могут оказаться несогласованными друг с другом, ввиду того, что каждое парное сравнение выполняется независимо от другого, возможно в другое время. В результате получаемая матрица парных сравнений оказывается несогласованной, и метод анализа иерархий не позволяет принять обоснованное решение о рангах альтернатив. При этом согласованность матрицы парных сравнений проверяется по мультипликативному признаку, предложенному Т. Саати вместе с фундаментальной шкалой предпочтений. Автором настоящего исследования предложен новый подход к численной интерпретации шкалы предпочтения альтернатив и оценке согласованности матрицы парных сравнений. В работе приводится процесс построения вероятностной шкалы предпочтений, а также описывается алгоритм оценки согласованности матрицы парных сравнений на основе статистических испытаний. Также приводится сравнение метода оценки согласованности матрицы парных сравнений, предложенного Саати, с методом, разработанным автором данной работы.

Ключевые слова и фразы: метод анализа иерархий; вероятностная шкала предпочтения; согласованность матрицы парных сравнений; алгоритм оценки согласованности; коэффициенты согласованности; имитационное моделирование.

1 ВВЕДЕНИЕ

В ситуациях многокритериального принятия решения в экономических и социальных системах возникают задачи, связанные с выбором лучших из альтернативных вариантов, распределением ресурсов меж-

ду альтернативными производителями и т.п. Метод анализа иерархий [1] позволяет найти лучший из альтернативных вариантов, распределить ресурсы между альтернативами пропорционально их приоритетам, а также может быть использован для сравнения сложных объектов в управлении качеством или в психолого-педагогическом тестировании [2,3].

Рассмотрим основные термины, которые будут встречаться в данной работе.

Главным процессом метода анализа иерархий является *парное сравнение*. Это сравнивание двух объектов по заданному признаку для установления степени предпочтения одного объекта другому.

При сравнении объектов степень предпочтения оценивается по *фундаментальной шкале предпочтений*, предложенной в [1] и используется в методе парных сравнений. После проведения метода парных сравнений полученные значения собираются в *матрицу парных сравнений*. Это квадратная A матрица размерности $(n\times n)$, элементы которой отражают результаты попарных сравнений n элементов. Элемент a_{ij} равен отношению предпочтения i-го объекта j-му объекту. Это отношение измеряется по фундаментальной шкале. Если при этом $a_{ij}=k$, то $a_{ij}=1/k$. Итак, матрица парных сравнений является *диагональной* $(a_{ii}=1)$ и *обратно симметричной* $(a_{ij}=1 / a_{ji})$ при $i,j=1,\ldots,n$.

Для возможности принятия решений по полученной матрице парных сравнений она должна быть согласованной. *Согласованность* – свойство матрицы парных сравнений, состоящее в том, что результаты сравнения i-го и j-го элементов, а также j-го и k-го элементов, позволяют получить результат сравнения i-го и k-го элементов. При отклонении от полной согласованности получение результата сравнения i-го и k-го элементов становится приблизительным.

Согласованность оценивается с помощью численного показателя отклонения матрицы парных сравнений от согласованного вида, называемого *индексом согласованности*.

Стохастический индекс согласованности – это статистический численный показатель отклонения матрицы парных сравнений от согласованного вида, характеризующий случайным образом составленную матрицу. Такой индекс зависит только от порядка матрицы парных сравнений.

Относительный индекс согласованности – это численный показатель отклонения матрицы парных сравнений от согласованного вида, оценивающий степень несогласованности матрицы по сравнению со стохастическим индексом несогласованности.

Рассматриваемый нами метод основывается на парных сравнени-

ях, в совокупности позволяющих ранжировать альтернативы на основе их попарного сравнения, выполняемого экспертами. Для попарного сравнения факторов автором метода Саати предложена специальная оценочная шкала, от 1 до 9, состоящая из 5 основных и 4 промежуточных суждений.

Степень предпочтения	**Определение**	**Комментарий**
1	Равна предпочтительность	Две альтернативы одинаково предпочтительны с точки зрения цели
3	Средняя степень предпочтения (слабо значимое)	Опыт эксперта позволяет считать одну из альтернатив немного предпочтительнее другой
5	Умеренно сильное предпочтение (значимое)	Опыт эксперта позволяет считать одну из альтернатив явно предпочтительней другой
7	Очень сильное предпочтение (очень значимое)	Опыт эксперта позволяет считать одну из альтернатив гораздо предпочтительней другой
9	Абсолютное предпочтение	Очевидно подавляющее предпочтение одной альтернативы другой

Таблица 1. Фундаментальная шкала предпочтений

При этом значения 2, 4, 6, 8 соответствуют промежуточным значениям уровня предпочтения.

Ввиду влияния различных «человеческих факторов» сравнения альтернатив могут оказаться несогласованными друг с другом – поскольку каждое парное сравнение выполняется независимо от другого, причём, возможно, в другое время. В результате получаемая матрица парных сравнений оказывается несогласованной, и метод анализа иерархий не позволяет принять обоснованное решение о рангах альтернатив. При этом согласованность матрицы парных сравнений проверяется по мультипликативному признаку, предложенному Саати одновременно с фундаментальной шкалой предпочтений.

2 ПРОТИВОРЕЧИЕ ФУНДАМЕНТАЛЬНОЙ ШКАЛЫ

В [1] согласованность парных сравнений описывается следующим образом: «В общем случае, под согласованностью подразумевается то, что при наличии основного массива необработанных данных все другие данные логически могут быть получены из них»[1]. Однако такое правило противоречит организации шкалы предпочтений, используе-

[1] Пунктуация приведена согласно [1] *(примечание редакции)*.

мой в методе анализа иерархий, что в свою очередь ограничивает возможности применения метода в некоторых технических и экономических задачах [4].

Для дальнейшего рассмотрения введём обозначения. Пусть значение предпочтения альтернативы *a* над альтернативой *b* равно *p*, тогда пишем $a \overset{p}{\succ} b$; значение предпочтения альтернативы *b* над альтернативой *c* равно *q*, пишем $b \overset{q}{\succ} c$; значение предпочтения альтернативы *a* над альтернативой *c* равно *r*, пишем $a \overset{к}{\succ} c$.

Предпочтения являются согласованными, если *r =f(p,q).*

Как несложно показать на основе изложенного выше, можно считать, что в [1] используется функция *f(p,q) = p*q*. При этом значения p,q изменяются в пределах основной шкалы от 1 до 9. Если же «предпочтение» оказывается обратным, то есть не альтернатива ***a*** предпочтительнее альтернативы ***b*,** а наоборот, то используется обратное значение величины предпочтения альтернативы *b* альтернативе ***a*,** то есть, если $b \overset{p}{\succ} a$, то $a \overset{\frac{1}{p}}{\succ} b$ [5]. Например, если $b \overset{4}{\succ} a$, то $a \overset{\frac{1}{4}}{\succ} b$. (Из этого правила и следует обратная симметричность матрицы парных сравнений А, задаваемая равенствами $a_{ij} = \dfrac{1}{a_{ji}}, \quad i, j = \overline{1, n}$.) При этом значения предпочтений альтернатив *a, b* и *c*, равные $a \overset{p}{\succ} b$, $b \overset{q}{\succ} c$, $a \overset{к}{\succ} c$, считаются согласованными, если *r=p*q*.

Например, при значении степени предпочтения альтернативы *a* альтернативе *b* равном 9, $a \overset{9}{\succ} b$, и значении степени предпочтения альтернативы *b* альтернативе *c* равном 8, $b \overset{8}{\succ} c$, значение согласованной степени предпочтения альтернативы *a* альтернативе *c* должно равняться 72, $a \overset{72}{\succ} c$, которое придется интерпретировать как «абсолютное предпочтение», $a \overset{9}{\succ} c$, что хотя и соответствует интуитивно «абсолютным предпочтениям» альтернативы *a* альтернативе *b* и альтернативы *b* альтернативе *c*, но не вписывается в заданную изначально шкалу степеней предпочтений от 1 до 9. Таким образом, получается, что исходная матрица не согласована и, вследствие этого, не может служить основой для принятия решения.

Пусть имеется n альтернатив $A_1, A_2, \ldots, A_{n-1}, A_n$. Обозначим $a_{ik} = Pr(A_i \succ A_k)$ степень предпочтения альтернативы A_i альтернативе A_k. Тогда для согласованности матрицы парных сравнений значения $a_{ik} = Pr(A_i \succ A_k)$ должны соответствовать таблице 2. Но значения в затененных частях таблицы, как указывалось выше, выходят за пределы исходной шкалы.

P(Ai≻AJ) = (Ai≻Ak) * P(Ak≻Aj)		**P(Ai≻Ak)**								
		1	2	3	4	5	6	7	8	9
P(Ak≻Aj)	**1**	1	2	3	4	5	6	7	8	9
	2	2	4	6	8	10	12	14	16	18
	3	3	6	9	12	15	18	21	24	27
	4	4	8	12	16	20	24	28	32	36
	5	5	10	15	20	25	30	35	40	45
	6	6	12	18	24	30	36	42	48	54
	7	7	14	21	28	35	42	49	56	63
	8	8	16	24	32	40	48	56	64	72
	9	9	18	27	36	45	54	63	72	81

Таблица 2. Согласованные предпочтения в соответствии со шкалой Саати.

В связи с приведёнными примерами возникает потребность построения более адекватной шкалы предпочтений и её интерпретации, соответствующей интуитивным представлениям о предпочтении, которое эта шкала будет моделировать.

3 ПОСТРОЕНИЕ ВЕРОЯТНОСТНОЙ ШКАЛЫ СТЕПЕНИ ПРЕДПОЧТЕНИЯ АЛЬТЕРНАТИВ

Чтобы преодолеть противоречие выхода за фундаментальную шкалу предпочтений, предлагается трактовать значение a_{ij} как ответ на вопрос «Какова вероятность того, что i-я альтернатива предпочтительнее j-й альтернативы?».

Рассмотрим связь вероятности того что A_i предпочтительнее A_k, то есть $P(A_i \succ A_k)$, вероятность того что A_k предпочтительнее A_j, то есть $P(A_k \succ A_j)$ и вероятность того что A_i предпочтительнее A_j, то есть $P(A_i \succ A_j)$?

Для этого требуется решить вероятностную задачу: пусть известно, что случайные числа *a,b,c* равномерно распределены в заданных интервалах на одной шкале и заданы вероятности *P(a>b)* и *P(b>c)*. Определить вероятность *P(a>c)*.

Решить данную задачу можно двумя способами. Первый способ – это аналитическое решение, которое вычисляется через геометрическую вероятность. Второй способ заключается в компьютерном имитационном моделировании геометрической вероятности, задавая случайные величины с равномерным распределением в определенных интервалах. Ввиду того, что вычисление геометрической вероятности требует громоздких математических вычислений, задача была решена с помощью второго способа – имитационного моделирования.

Решить задачу моделирования геометрической вероятности для заданных значений *P(a>b)* и *P(b>c)* можно с помощью имитационного моделирования [2]. Для этого достаточно, смоделировать случайные величины с равномерными распределениями в таких интервалах, которые обеспечивают вероятности *P(a>b)* и *P(b>c)*. Затем произвести достаточно большое число испытаний для получения репрезентативной выборки, и вычислить приближенное значение вероятности *P(a>c)* как частоту события *(a>c)* в полученной выборке.

При построении имитационной модели необходимо учитывать, что интервалы для случайных величин *a*, *b* и *c* выбираются в зависимости от значений $p = P(a>b)$ и $q = P(b>c)$.

Отметим, что приведенный пример вычислен с определенной степенью точности, но ничто не препятствует вычислению согласованных степеней превосходства и для дробных значений шкальных оценок

Результаты имитационного моделирования показывают адекватность использования вероятностной шкалы, результаты которого можно увидеть в таблице 3.

P(Ai≻AJ)		**P(Ai≻Ak) (p(a>b) = p)**								
		0.5	0.5625	0.625	0.6875	0.75	0.8125	0.875	0.9375	1.0
P(Ak≻Aj) (p(b>c) = q)	0.5	0.49	0.56	0.62	0.68	0.74	0.81	0.87	0.93	0.99
	0.5625	0.56	0.61	0.67	0.72	0.78	0.83	0.89	0.94	0.99
	0.625	0.62	0.67	0.71	0.76	0.81	0.85	0.9	0.95	0.99
	0.6875	0.68	0.72	0.76	0.8	0.84	0.88	0.92	0.96	0.99

	0.75	0.75	0.78	0.81	0.84	0.87	0.9	0.93	0.97	1.0
	0.8125	0.81	0.83	0.85	0.88	0.9	0.92	0.95	0.97	1.0
	0.875	0.87	0.89	0.9	0.92	0.93	0.95	0.96	0.98	1.0
	0.9375	0.93	0.94	0.95	0.96	0.96	0.97	0.98	0.99	1.0
	1.0	0.99	0.99	0.99	0.99	1.0	1.0	1.0	1.0	1.0

Таблица 3 Согласованные предпочтения
в соответствии с вероятностной шкалой

В таблице 4 можно увидеть предложенную вероятностную шкалу которая получилась путем деления вероятности от 0,5 до 1 на 8 равных частей. Если объекты равнозначны, то предпочтение одного другому проявляется с вероятностью 0,5. Если объект А предпочтительнее В с оценкой 9, то предпочтение одного другому проявляется с вероятностью 1. Промежуточные значения шкалы нарастают равномерно по 0,125 на каждый балл оценки предпочтения.

Фундаментальная шкала	1	2	3	4	5	6	7	8	9
Вероятностная шкала	0.5	0.5625	0.625	0.6875	0.75	0.8125	0.875	0.9375	1.0

Таблица 4. Вероятностная шкала

4 МЕТОД ОЦЕНКИ СОГЛАСОВАННОСТИ ДЛЯ ВЕРОЯТНОСТНОЙ ШКАЛЫ ПРЕДПОЧТЕНИЙ

После получения согласованных уровней предпочтений возникает проблема согласованности экспертных оценок, так как алгоритм оценки согласованности матрицы парных сравнений, предлагаемый Саати, построен для мультипликативной функции и не подходит для вероятностной шкалы.

Поэтому требуется новая мера рассогласованности матрицы парных сравнений, в качестве которой предлагается использовать квадратичную меру различия матриц $\|A-B\| = \sqrt{\sum_{i,j}(\ln(a_{ij})-\ln(b_{i\,j}))^2}$, вычисленную для исходной матрицы парных сравнений А и полностью согласованной матрицы В, полученной по *(n-1)* элементам матрицы А. В данной формуле были использованы натуральные логарифмы[4], так как фундаментальная шкала Саати имеет схожее поведение.

Тогда индекс рассогласованности C_I будет определяться формулой

$$C_I = ||AP - AP^*||,$$

где:

AP – матрица парных сравнений полученная по вероятностной шкале; *AP** – полностью согласованная матрица, полученная по матрице *AP*, редуцированной до 1-й строки.

Стохастический индекс рассогласованности R_I вычислен как усреднённое значение меры рассогласованности – при случайно заполнении МПС с сохранением её асимметричности.

Таким образом, многократно задавая «случайную матрицу» *AP* заданного порядка *n*, с элементами из интервала (0,1) и удовлетворяющими аддитивной антисимметрии ($ap_{ij}=1—ap_{ji}$), редуцируя её до 1-й строки, вычисляя полностью согласованную матрицу *AP** , а затем – индекс рассогласованности C_I, можно получить достаточно большую выборку дающую «устойчивое» среднее значение индекса рассогласования при большом количестве испытаний. Такое среднее значение и будет «стохастическим индексом рассогласованности», аналогичным индексу R_I в алгоритме Саати.

5 АНАЛИЗ МЕТОДОВ ИНТЕРПРЕТАЦИИ СТЕПЕНИ ПРЕДПОЧТЕНИЯ ПРИ ИСПОЛЬЗОВАНИИ РАЗЛИЧНЫХ ШКАЛ

Заключительный шаг в определении степени рассогласованности вероятностной матрицы парных сравнений *AP* аналогичен шагу алгоритма Саати. А именно, в качестве «относительного индекса рассогласованности» матрицы AP берётся отношение величин C_I и R_I [6]:

$$C_R = \frac{C_I}{R_I}.$$

Автором было проведено сравнение графиков метода расчета оценки рассогласованности матрицы парных сравнений, предложенного автором статьи, и метода Саати (см. рис. 1 ниже).

Сопоставление графиков изменения C_R показывает сходное поведение функции относительного рассогласования при вычислении отклонения от согласованной матрицы по метрике, используемой Саати (через максимальное собственное число матрицы) и по естественной (евклидовой) метрике. При *n*=3, случайным образом выбирается только один элемент матрицы *AP*, который и отражается на оси абсцисс. При заданных значениях элементов первой строки матрицы парных сравнений, полностью согласованным значением для ap_{23} является число 1,5. Синий график «логарифмически симметричен» относительно согласованного значения 1,5 (слева от 1,5 степень рассогласования становится отрицательной, если отразить значения относитель-

но оси OX), а красный график станет симметричным относительно значения 1,5, если левую часть растянуть в 9 раз.

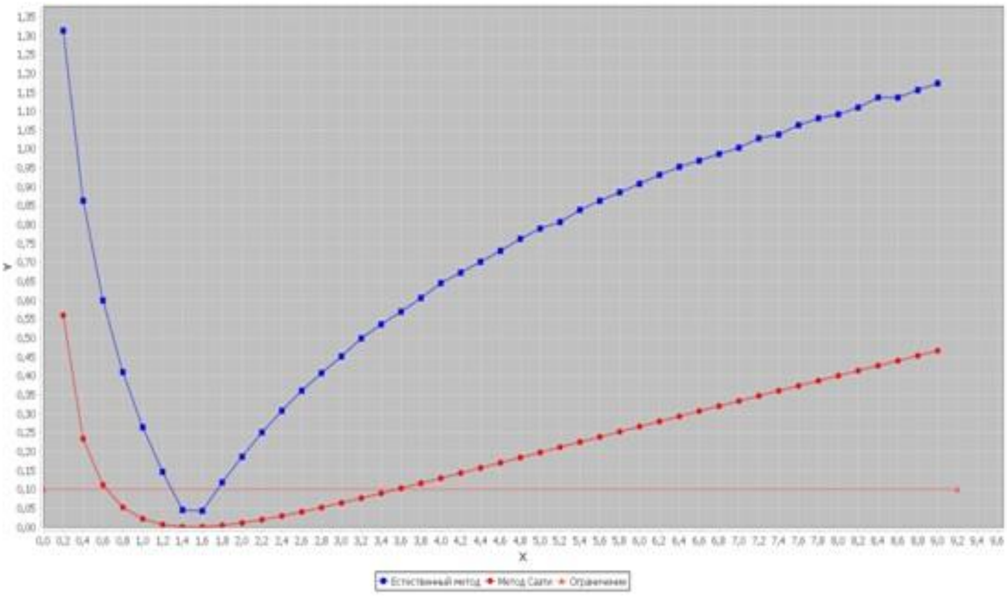


Рисунок 1. Сравнение графиков метода Саати и естественного метода (*n*=*3*)

Это позволяет сделать вывод, что можно использовать и «естественный» метод для оценки рассогласованности матриц парных сравнений. Для этого остается лишь определить интервал допустимых отклонений C_I от 0 (полная согласованность).

В алгоритме Саати задана допустимая граница для индекса рассогласованности $C_I < 0.1$, Такому ограничению удовлетворяют значения a_{23} от 0,64 до 3,56 (по фундаментальной шкале). Как видно из графика на рис. 1, в случае использования «вероятностной шкалы», допустимой границей становится $C_I < 0,55$.

6 ВЫВОДЫ

Мной была разработана вероятностная шкала предпочтений для метода анализа иерархий, которая позволяет преодолеть проблему противоречивости мультипликативной интерпретации предпочтения, состоящую в выходе оценок за границы фундаментальной шкалы Саати.

Разработан и описан универсальный метод, который дает возможность оценивать согласованность матрицы парных сравнений составленных как с помощью фундаментальной шкалы Саати, так и с помощью вероятностной шкалы.

СПИСОК ЛИТЕРАТУРЫ

1. Саати, Т. (1993), *Принятие решений: метод анализа иерархий*,

Радио и связь, М., 320 с.
2. Ярыгин, А. Н., Ярыгин, О. Н. (2012), *Лекции и задачи по дискретной математике (от теории к алгоритмам)*, Издательство «Тонкие наукоемкие технологии», Старый Оскол, 392 с.
3. Ярыгин, А. Н., Ярыгин, О. Н. (2011), «Относительное ранжирование интеллектуальных компетентностей с помощью интерактивных парных сравнений», *Вектор науки Тольяттинского государственного университета*, № 2 (16), сс. 413–417.
4. Ярыгин, О. Н., Беляев, М. А., Темирджанова, М. А. (2014), «Принятие управленческий решений в многокритериальных задачах на основе вероятностной матрицы парных сравнений», *Карельский научный журнал*, № 4 (9), сс. 98–100.
5. Тутыгин, А. Г., Коробов, В. Б. (2010), «Преимущества и недостатки метода анализа иерархий», *Известия Российского государственного педагогического университета им. А. И. Герцена*, № 122, сс. 108–115.
6. Ярыгин, О. Н., Беляев, М. А. (2013), «Уточнение вида функции предпочтения альтернатив в методе анализа иерархий», *Карельский научный журнал*, № 4 (5), сс. 49–52.

Probabilistic approach to solve multicriteria problems using analytic hierarchy process

M. A. Belyaev, email: mbelyaev1991@gmail.com
Samara State Aerospace University

Abstract. This work examine method of analytic hierarchy process and problems of output of the matrix paired comparisons as fundamental scale of preferences. Analytic hierarchy process allow find best of alternative options, allocate resources between alternative options in proportion of their priority. Also could be used for compare difficult objects in quality control or psycho-pedagogical testing. This method based on paired comparisons, in the aggregate allow rank alternatives based on pairwise comparisons performed by experts. Due to the impact of various "human factors" comparison of alternatives might turn up inconsistent with each other since each paired comparison performed independent from other or possible at another time. As result, received matrix paired comparisons appear inconsistent, and analytic hierarchy process can't allow accept reasonable decision of ranks of alternatives. Wherein consistency of matrix paired comparisons tested on the basis of the multiplicative attribute, proposed by T. Saaty fundamental scale of preferences. Author of present research offer new approach to numerical interpretation scale of preferences of alternatives and evaluate of consistency of matrix paired comparisons. This work present the process of building a probability scale of preferences, and also describes algorithm for estimating the coherence matrix of pairwise comparisons based on statistical tests. In addition, present comparison method evaluate of consistency of matrix-paired comparisons proposed by Saaty with method developed by author of this work.

Key words and phrases: analytic hierarchy process; probability scale of preference; consistency of matrix-paired comparisons; algorithm for consistency of matrix-paired comparisons; mismatch coefficients; simulation modeling.

Computing Classification System 1998: H.4.2

Mathematics Subject Classification 2010: 90B50

REFERENCES

1. Saaty, T. (1993), *Decision making: analytic hierarchy process* [*Prinyatie reshenij: metod analiza ierarhij*] Moscow: Radio i svyaz, 1993, 320 p.
2. Yarigin, A. N., Yarigin, O. N. (2012), *Lectures and problem in discrete mathematics (from theory to algorithms)* [*Lekcii i zadachi po diskretnoj matematike (ot teorii k algoritmam)*] TNT Publishing, Stariy Osrol, 392 p.
3. Yarigin, A. N., Yarigin, O. N. (2011), *"Relative ranking intellectual competences-by interactive paired comparisions", Vector Science Togliatti State University [“Otnositelnoe ranzhirovanie intellektualnyh kompetent nostej s pomoshchyu interaktivnyh parnyh sravnenij", Vektor nauki Tolyattinskogo gosudarstvennogo universiteta*], No. 2 (16), pp. 413–417.
4. Yarigin, O. N., Belyaev, M. A., Temirdzhanova, M. A. (2014), *"Management decisions in multicriterion tasks based on a probability matrix of paired comparisons", Karelian Scientific Journal* [*"Prinyatie upravlencheskij reshenij v mnogokriterialnyh zadachah na osnove veroyatnostnoj matricy parnyh sravnenij", Karelskij nauchnyj zhurnal*], No. 4 (9), pp. 98–100.
5. Tutygin, A. G., Korobov, V. B. (2010), *"Advantages and desadvantages of the analytic hierarchy process", Izvestia: Herzen University Journal of Hu-*

manities & Science [*“Preimushchestva i nedostatki metoda analiza ierarhij”, Izvestia Rossijskogo gosudarstvennogo pedagogicheskogo universiteta im. A.I. Gercena*], No. 122, pp. 108–115.

6. Yarigin, O. N., Belyaev, M. A. (2013), “*Clarifying of preference function in the analytic hierarchy process”, Karelian Scientific Journal* [*“Utochnenie vida funkcii predpochteniya alternativ v metode analiza ierarhij”, Karelskij nauchnyj zhurnal*], No. 4 (5), pp. 49–52.

УДК 004.8

МОДЕЛЬ МНОГОУРОВНЕВОГО АНАЛИЗА ТЕКСТА НА ОСНОВЕ ПОИСКА ВГЛУБЬ В ПРОСТРАНСТВЕ ИНТЕРПРЕТАЦИЙ *

Е. И. Большакова, email: eibolshakova@gmail.com
А. А. Носков, email: alexeynoskov@gmail.com
Московский государственный университет им. М. В. Ломоносова

Аннотация. В работе предлагается модель анализа текста на естественном языке, учитывающая общепринятое разбиение процесса анализа на этапы (графематический, морфологический, синтаксический, семантико-прагматический), но не требующая промежуточных вспомогательных этапов для сокращения неизбежно возникающих вследствие омонимии множественных вариантов анализа единиц текста (символов, слов, словосочетаний, предложений). Результаты всех этапов (уровней) анализа представляются в модели единообразно, в виде графов интерпретаций текста, что позволяет в компактной форме хранить все многочисленные варианты анализа. Ключевой особенностью предлагаемой модели является одновременное проведение анализа текста на нескольких уровнях, при этом более глубокие этапы обработки текста управляют предшествующими, запрашивая у них интерпретацию (вариант анализа) очередной единицы текста и выполняя затем проверку корректности этой интерпретации относительно текущего уровня и построение интерпретации этого уровня. В случае некорректности интерпретации, полученной с предыдущего этапа, и соответственно, невозможности построить интерпретацию текущего уровня, для продолжения анализа выполняется запрос другой интерпретации предшествующего уровня. Тем самым возможен полный просмотр пространства интерпретаций единиц текста на всех его уровнях.
Поскольку рассматриваемая модель реализует поиск вглубь в пространстве интерпретаций, построение интерпретации наиболее глубокого уровня возможно уже для начальных фрагментов текста, без проведения полного анализа текста. В работе описываются базовые операции над графами интерпретаций текста и необходимая организация процедур анализа каждого уровня. Характеризуется также общая схема многоуровневой обработки текста, указываются особенности ее реализации на функционально-объектном языке программирования с использованием «ленивых» вычислений.

Ключевые слова и фразы: анализ текста на естественном языке; уровни и этапы анализа текста; пространство интерпретаций текста; графы интерпретаций; модель многоуровневого анализа текста.

---

1. ВВЕДЕНИЕ

В последние годы растет количество программных приложений, требующих автоматической обработки текста на естественном языке (АОТ). Чаще всего анализ неструктурированного текста требуется в системах информационного поиска, вопросно-ответных системах, системах машинного перевода, при решении задач реферирования и аннотирования текстов, извлечения информации из текстов.

Процесс анализа текста включает обычно несколько последовательных этапов [1], в том числе графематический анализ (выделение в тексте слов-лексем, или токенов), морфологический анализ выделенных словоформ, синтаксический анализ предложений и словосочетаний, а также этап семантико-прагматического анализа. Отметим, что количество выполняемых этапов анализа зависит от решаемой прикладной задачи, а последовательность их выполнения обычно соответствует уровням обработки текста – от поверхностных к более глубоким.

Одной из серьёзных проблем, касающихся всех этапов анализа, является проблема неоднозначности – как правило, на каждом этапе возможно несколько вариантов анализа (интерпретаций) единиц текста, причём выбор правильного варианта в общем случае не может быть сделан без привлечения информации, получаемой на более глубоких уровнях анализа. В частности, разрешение такой неоднозначности, как морфологическая омонимия (совпадение различных форм одного или нескольких разных слов: например, *мою* – это форма глагола *мыть* или же местоимения *мой*) обычно осуществляется с учётом локальных синтаксических связей слов, т.е. информации уровня синтаксического анализа.

В действующих системах АОТ проблема неоднозначности обычно решается путем введения дополнительных, промежуточных этапов анализа, на которых отбрасывается часть вариантов анализа за счёт опережающего применения отдельных проверок, относящихся к более глубоким уровням анализа. Например, дополнительный этап постморфологической обработки служит для разрешения морфологической омонимии на основе выявления некоторых видов синтаксических связей. В целом такой путь усложняет общую схему анализа, делает его менее эффективным из-за дублирования отдельных операций (проверок), и в то же время не решает проблему полностью, так как оставляет часть некорректных интерпретаций, отбрасываемых только на последующих этапах анализа.

В данной работе предлагается модель анализа текста на естественном языке, учитывающая общепринятое разбиение процесса анализа на

этапы и не требующая введения вспомогательных промежуточных этапов для сокращения получающихся вариантов анализа. Результаты каждого этапа анализа представляются в модели единообразно, в виде графов интерпретаций [2], позволяющих в компактной форме хранить все множество вариантов анализа. Построение интерпретации наиболее глубокого уровня возможно уже для начальных фрагментов текста – за счёт организации поиска вглубь в пространстве интерпретаций всех уровней анализа.

Некоторые аспекты предлагаемой модели близки к идеям поточной обработки текста и учёта многовариантности анализа, высказанным в работах [3–5]. В системах LT-NSL и Wraeltic для реализации различных этапов обработки используются параллельные процессы операционной системы, передающие данные друг другу последовательно, что позволяет обрабатывать большие тексты, не храня их полностью в памяти. В проекте системы CAFE [5] процессы анализа более глубокого уровня предоставляют предыдущим уровням обратную связь, влияющую на отбор интерпретаций на предыдущих уровнях. Также необходимо отметить работу [6], где был предложен эффективный алгоритм синтаксического анализа с учётом неоднозначности. Предлагаемая нами модель использует схожие элементы, однако направлена не на один этап анализа, а на интеграцию различных этапов.

В рассматриваемой модели многоуровневого анализа текста более глубокие этапы анализа управляют предшествующими, получая от них результаты – интерпретации фрагментов текста – и осуществляя их отбор. Модель реализована на функционально-объектном языке программирования Scala [7]. Особенностью функциональной реализации является использование т.н. ленивых вычислений, в частности, «ленивая реализация» графов интерпретаций [8]. В недавних работах [9, 10] ленивые вычисления применяются для реализации синтаксического анализа с учётом многовариантности деревьев разбора, в нашей же модели ленивая реализация характерна для всех этапов анализа.

2. ЭТАПЫ АНАЛИЗА И ИНТЕРПРЕТАЦИИ

Общепринятым подходом к организации анализа текстов на естественном языке является выделение нескольких последовательных этапов, каждый из которых решает задачи анализа текста на одном из его уровней. К основным уровням относятся

- уровень символов,
- уровень слов (словоформ),
- уровень предложений (синтаксических конструкций),
- уровень высказываний.

Каждый этап, в том числе графематический, морфологический, синтаксический анализ реализует переход с одного уровня на следующий, более глубокий.

Важно, что при таком подходе различные этапы анализа могут быть реализованы с помощью независимо разработанных программных модулей, что значительно снижает сложность построения систем АОТ. В то же время жёсткая последовательность применения этапов к обрабатываемому тексту ограничивает решение проблемы неоднозначности анализа, неизбежно возникающей на каждом из уровней.

Будем называть интерпретацией некоторой единицы текста или всего текста один из вариантов их анализа на каком-либо уровне. Множество интерпретаций текста на графематическом уровне включает всевозможные способы его разбиения на лексемы (токены). Интерпретации на морфологическом уровне охватывают различные варианты анализа словоформ текста, более точно – всевозможные сочетания этих вариантов, например, фрагмент "*Я мою пол на кухне*" имеет 24 различных интерпретации (*я* – один вариант анализа, *мою* – два варианта, соответствующих глаголу и местоимению, *пол* – три: существительное в двух разных падежах и краткое прилагательное, *на* – два: частица и предлог, *кухне* – два, соответствующих разным падежам). На синтаксическом уровне интерпретации включают различные синтаксические деревья для одного фрагмента текста.

Проблему составляет именно множественность интерпретаций текста, поскольку отбор правильной интерпретации из всего множества допустимых на текущем уровне может быть сделан в общем случае только на следующих уровнях анализа. Передача же на следующий этап всего множества интерпретаций значительно затрудняет анализ, особенно для случаев больших текстов (количество интерпретаций часто экспоненциально растет с увеличением длины текста).

Для сокращения числа интерпретаций, передаваемых на следующий этап анализа, часто вводятся дополнительные, промежуточные этапы анализа. Так, например, синтаксический анализ часто предваряется этапом постморфологического анализа, призванного разрешить морфологическую омонимию. В целом такой подход с введением промежуточных этапов анализа имеет ряд недостатков. Во-первых, введение дополнительных проверок усложняет общую схему анализа и делает её менее прозрачной. Во-вторых, фильтрация интерпретаций средствами более простыми, чем полный анализ на следующих уровнях, способна отсеять верные интерпретации, которые не были бы далее отброшены, и в то же время оставить интерпретации, некорректность которых определилась на следующих этапах.

В предлагаемой нами модели анализа текста можно обойтись без введения промежуточных этапов анализа за счёт трёх следующих положений.

- Анализ текста и его единиц ведётся сразу на нескольких уровнях и завершается построением некоторой интерпретации наиболее глубокого уровня, не дожидаясь полного завершения анализа всего текста (и рассмотрения всех возможных интерпретаций).
- Каждый этап анализа управляет предшествующим этапом, запрашивая у него очередной элемент интерпретации и осуществляя затем его обработку, включая проверку на корректность и построение соответствующего элемента интерпретации следующего уровня.
- В каждый момент происходит построение одной интерпретации; в случае невозможности продолжить её построение происходит запрос другой интерпретации предшествующего уровня для продолжения анализа. Тем самым возможен полный просмотр всего пространства интерпретаций.

Указанные свойства модели обеспечиваются, во-первых, используемым на каждом уровне анализа внутренним представлением данных (интерпретаций текста) в виде графа интерпретаций, и, во-вторых, определённой внутренней организацией процедур-этапов анализа. Граф интерпретаций каждого уровня позволяет хранить множество интерпретаций в компактном виде и проводить анализ только части из них. Внутренняя организация процедур-этапов требует, чтобы они позволяли получать результаты анализа текста по частям, не дожидаясь завершения предыдущей процедуры и рассмотрения всех интерпретаций.

Рассматриваемая модель в определенной степени моделирует анализ текста человеком, который по мере прочтения текста сразу строит некоторую интерпретацию его смысла, не дожидаясь его окончания. При этом, если очередной фрагмент текста не согласуется с уже построенной интерпретацией, то делается попытка найти другую интерпретацию, соответствующую всему прочитанному на данный момент тексту.

Отличия предлагаемой модели от общепринятого подхода к обработке текста по уровням касается передачи данных и управления между этапами анализа: в общепринятом подходе управление и данные передаются последовательно от этапа к этапу, в то время как *в нашей модели управление передаётся от глубоких уровней к поверхностным, а данные – в обратном направлении*.

3. ГРАФЫ ИНТЕРПРЕТАЦИЙ

В качестве внутренней структуры данных для хранения обрабатываемой информации на каждом из уровней анализа текста используется нагруженный ориентированный ациклический граф, называемый графом интерпретаций.

Вершины графа соответствуют определённым позициям анализируемого текста, а метки рёбер представляют элементы, из которых состоит интерпретация этого текста на рассматриваемом уровне. В таком графе каждый путь от истока к стоку соответствует некоторой интерпретации анализируемого текста.

Граф интерпретаций позволяет компактно представлять множество интерпретаций с учётом часто встречающейся локальности неоднозначностей. Локальная неоднозначность имеет место, когда различия между двумя интерпретациями текста касаются только одного сравнительно небольшого его фрагмента. За счёт возможности разветвлений и объединений путей в графе обеспечивается возможность хранить общие участки различных интерпретаций только один раз, что позволяет значительно уменьшить требуемый объём памяти.

Например, все 24 морфологические интерпретации приведённого выше фрагмента «Я мою пол на кухне» представлены в графе, показанном на рис. 1. Общее число рёбер этого графа равно сумме количеств вариантов анализа входящих в текст слов, что значительно меньше, чем их произведение (количество путей в графе).

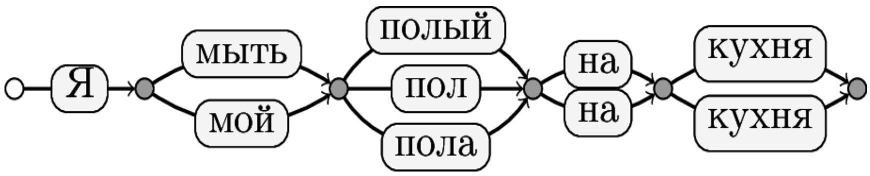


Рис. 1. Пример графа интерпретаций

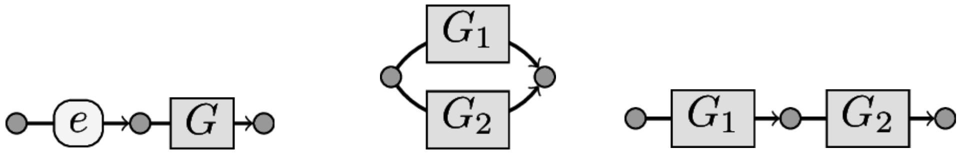
В рассматриваемой модели анализа текста множество используемых графов интерпретаций каждого уровня включает тривиальный граф G_ε из одной вершины, исток и сток которого совпадают – такой граф представляет множество из одной пустой интерпретации (соответствующей пустому фрагменту текста).

На множестве графов интерпретаций определены следующие базовые операции:

- операция присоединения ребра e к истоку графа G – см. рис. 2а. В результирующем графе $e.G$ общее число интерпретаций не меняется, но к каждой из них приписывается начальный элемент e;
- операция объединения графов интерпретаций G_1 и G_2 – см.

рис. 2б. Результирующий граф $G_1 \cup G_2$ представляет объединённое множество интерпретаций, полученное путём склеивания истоков и стоков исходных графов.

На основе композиции этих базовых операций может быть определена также операция конкатенации графов – см. рис. 2в. В результате этой операции из двух графов интерпретаций G_1 и G_2 образуется новый граф $G_1.G_2$, представляющий множество интерпретаций (путей), каждая из которых есть конкатенация некоторой интерпретации (пути) из первого графа и интерпретации (пути) из второго графа.



а) Присоединение б) Объединение в) Конкатенация

Рис. 2. Операции над графами интерпретаций

Любой граф интерпретаций может рассматриваться как результат определённой композиции рассмотренных операций. В частности, изображённый на рис. 1 граф может быть получен путём конкатенации графов G_1, ..., G_5, каждый из которых представляет множество различных вариантов анализа соответствующей словоформы текста (*я*, *мою*, *пол*, *на*, *кухне*); эти графы, в свою очередь, получаются в результате объединения графов, состоящих из одного ребра e_i (варианта анализа i-й словоформы).

По сравнению с известными алгебрами графов, например [11], в нашей модели используется упрощенная система графов нужного нам вида.

4. ОРГАНИЗАЦИЯ ПРОЦЕДУР АНАЛИЗА НА ГРАФАХ

Поскольку информация на различных уровнях анализа представляется графами интерпретаций, любая процедура, реализующая этап анализа, действует как преобразование входного графа интерпретаций одного уровня в выходной граф интерпретаций следующего уровня. В нашей модели каждая процедура анализа должна удовлетворять двум условиям:

1. Для того чтобы разные интерпретации, представленные во входном графе, можно было обрабатывать независимо друг от друга, необходимо допускать применение процедуры к подмножествам всего множества входных интерпретаций с последующим объединением её результатов в выходном графе интерпретаций.

2. Для того чтобы частичные результаты текущего этапа анализа можно было передавать на следующий уровень анализа, не дожидаясь завершения этого этапа, процедура должна вырабатывать элементы интерпретаций следующего уровня по мере получения элементов интерпретаций из входного графа.

Будем рассматривать каждую процедуру анализа как процесс *F(S,G)*, обладающий внутренним состоянием *S*, в котором он последовательно обрабатывает входной граф интерпретаций *G*. Важно, что процесс *F* вырабатывает фрагменты выходного графа интерпретаций (подграфы), не дожидаясь завершения обработки всего входного графа.

Действие процесса анализа определяется по шагам: на каждый шаг поступает элемент интерпретации (ребро) входного графа; выдаётся некоторый подграф результирующего графа, и при этом происходит смена состояния процесса. Однако, в силу того, что в процессе анализа могут возникать неоднозначности, на очередном шаге обработки процесс может ветвиться, выдавая несколько разных пар «выходной подграф – новое состояние». Получающиеся в результате ветвления процессы аналогичны используемым в алгоритме GLR [6], но не оперируют стеком и имеют чётко определенный выход в виде фрагментов графа интерпретаций, поступающего на вход следующего этапа анализа.

Применительно к одному шагу процесса анализа первое из указанных выше необходимых условий работы может быть записано как

$$F(S, G_1 \cup G_2) = F(S, G_1) \cup F(S, G_2)\,, \tag{1}$$

что соответствует обработке процессом *F* в состоянии *S* входного графа, являющегося объединением двух других графов.

Следующая формула описывает второе из указанных необходимых условий с учётом возможности ветвления: применение шага процесса *F* в состоянии *S* к графу с выделенным первым ребром *e* может быть выражено

$$F(S, e.G) = \bigcup (G_i.F(S_i, G))\,. \tag{2}$$

Здесь процесс выполняет обработку входного графа, являющегося результатом присоединения ребра *e* (элемента интерпретации) к некоторому подграфу *G*. Знак теоретико-множественного объединения в правой части формулы означает ветвление процесса в силу неоднозначности обработки (интерпретации) этого элемента *e*. В общем случае $G_i.F(S_i,G)$ соответствует *i*-й ветви процесса, которая вычисляется как конкатенация графа G_i, являющегося выходом текущего шага, и графа, вырабатываемого процессом *F* в новом состоянии S_i с входным графом

G (оставшаяся часть исходного графа). Структура входного и выходного графа интерпретаций в рассматриваемом случае изображена на рис. 3. Важно, что на любой ветви процесса граф G_i является подграфом выходного графа интерпретаций и (если он не пуст) может быть сразу передан на следующий этап анализа.

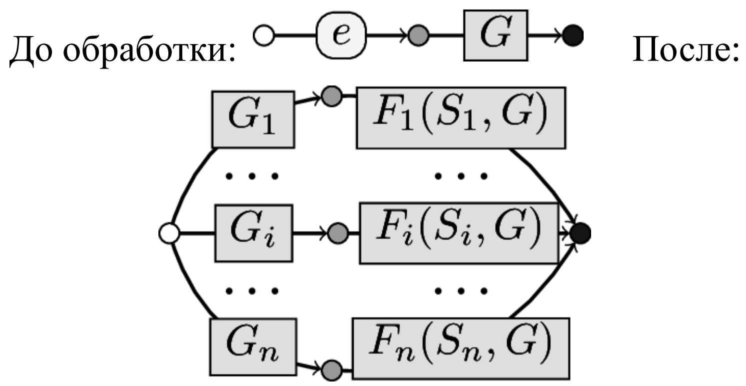


Рис. 3. Ветвление на шаге обработки ребра графа

Для полноты определения процесса F действие его шага должно быть задано на тривиальном графе G_ε, представляющем пустую интерпретацию:

$$F(S, G_\varepsilon) = G, \tag{3}$$

что означает завершение процесса F при обработке графа интерпретаций G_ε.

Отметим, что в частном случае процесс, реализующий некоторый этап анализа, может не требовать смены своего внутреннего состояния. Например, действие этапа морфологического анализа состоит в преобразовании каждой единицы текста (словоформы) в группу её морфологических интерпретаций. Поскольку результат каждого шага (обработка очередной словоформы) не зависит от других, состояние процесса остается неизменным.

Ещё одна возможность – пустой выход (пустой граф) на некотором шаге процесса, при смене его внутреннего состояния. Так, при проведении графематического анализа на базе конечного автомата на каждом шаге процесс получает на вход символ исходного текста, однако выдаёт элементы интерпретации (выделенные лексемы) не на каждом шаге, но при этом всегда происходит смена состояния процесса (что соответствуют накоплению автоматом выделяемой лексемы).

5. СХЕМА МНОГОУРОВНЕВОЙ ОБРАБОТКИ ГРАФОВ

В соответствии с предлагаемой моделью, многоуровневый анализ текста представляет из себя постепенное преобразование графов интерпретаций на всех уровнях вышеописанными процессами. Поскольку процесс каждого уровня требует в качестве входного графа элементы интерпретаций предыдущего уровня, то в случаях, когда требуемые входные данные еще не вычислены, процесс приостанавливается и передает управление процессу предыдущего уровня, который их вычисляет.

Таким образом, в каждый момент текущее состояние анализа текста включает:

- фрагменты G_k графов интерпретаций различных уровней, которые еще не были проанализированы и ожидают анализа;
- приостановленные процессы анализа F_j с внутренними состояниями S_j, ожидающие вычисления графов интерпретаций G_j предыдущих уровней.

Перед началом анализа текста в число непроанализированных фрагментов графов интерпретаций входит граф, соответствующий исходному уровню символов текста, а также тривиальные графы более глубоких уровней (которые пока еще не содержат рёбер). При этом приостановленные, а точнее, ещё не начатые процессы анализа этих графов ожидают вычисления тривиальных графов предшествующих уровней.

Обработка графов интерпретаций происходит в следующем цикле.

1. На одном из уровней анализа выбирается один из приостановленных процессов – согласно одной из возможных стратегий выбора.
2. Если входной граф выбранного процесса ещё не вычислен, то выбранный процесс откладывается и управление передаётся соответствующему приостановленному процессу для вычисления этого входного графа.
3. Если же входной граф выбранного процесса уже вычислен, то выполняется шаг этого процесса, результат которого зависит от вида входного графа:
   - если входной граф – объединение других графов, то шаг реализуется по формуле (1);
   - если у входного графа выделено первое ребро, то обработка производится в соответствие с формулой (2): в результате обработки этого ребра к выходному графу в общем случае добавляется несколько подграфов G_i, а также порождаются процессы $F(S_i,G)$ того же уровня с внутренним состоянием S_i, ожидающие вычисления фрагмента G входного графа;

- если входной граф является тривиальным, то шаг процесса реализуется по формуле (3), т.е. процесс завершается, выдавая последний фрагмент графа текущего уровня.

4. Цикл обработки графов интерпретаций заканчивается, когда в графе интерпретаций самого глубокого уровня построен полный путь из истока в сток (представляющий искомую интерпретацию анализируемого текста); в ином случае происходит переход на начало цикла.

Если же требуется найти большее число интерпретаций текста, то цикл обработки может быть возобновлён – до тех пор, пока не будет получено необходимое число интерпретаций самого глубокого уровня, или же все процессы не будут завершены, что означает, что проанализированы все возможные интерпретации на всех уровнях.

Таким образом, на каждом шаге рассмотренного цикла реализуется шаг работы одного из приостановленных процессов обработки графов интерпретаций. Процессы анализа постепенно перерабатывают фрагменты графов интерпретаций различных уровней, порождая фрагменты графов более глубоких уровней.

Возможные стратегии выбора приостановленного процесса для продолжения анализа, подразумевают, во-первых, выбор этапа анализа. Выбор процесса с наиболее глубокого уровня анализа обеспечивает более быстрое построение законченных интерпретаций текста (интерпретаций последнего уровня). В ином случае может происходить накопление фрагментов графов промежуточного уровня, многие из которых могут быть ошибочными и были бы отсеяны на более глубоких уровнях.

Во-вторых, возможен различный выбор одного из приостановленных процессов в рамках последнего этапа (наиболее глубокого уровня), а именно:

- в случае выбора процесса с входным графом, расположенным наиболее далеко от истока графа интерпретаций рассматриваемого уровня, мы имеем стратегию, аналогичную поиску в глубину, при которой продолжается обработка текущей интерпретации; именно эта стратегия используется в нашей модели анализа текста;
- в случае выбора ребра, расположенного наиболее близко к истоку графа, мы получаем стратегию, аналогичную поиску в ширину – что позволяет обрабатывать все интерпретации наиболее глубокого уровня одновременно.

Ниже на рис. 4 показано промежуточное состояние анализа фразы *«Я мою пол на кухне»* в момент, когда процессами морфологического уровня были затребованы с графематического уровня и обработаны три

первые словоформы (при этом они удалены из графа интерпретаций графематического уровня). Часть найденных на морфологическом уровне интерпретаций также уже обработано процессами синтаксического уровня: обработаны (и удалены с морфологического уровня) интерпретация слова *я*, одна морфологическая интерпретация словоформы *мою* (глагол) и одна интерпретация словоформы *пол* (форма прилагательного *полый*). В результате построено частичное синтаксическое дерево для словосочетания *я мою*. При этом синтаксический анализ ведётся согласно формальной грамматике

$$
\begin{aligned}
&S \to NG\ \ VG \\
&NG \to N \mid PR \\
&VG \to V\ \ NG\ \ \mid VG\ \ \ PP\ NG\,,
\end{aligned}
$$

где нетерминалы S, NG, VG, N, PR, V, PP означают соответственно: предложение, группу существительного, группу глагола, существительное, местоимение, глагол, предлог.

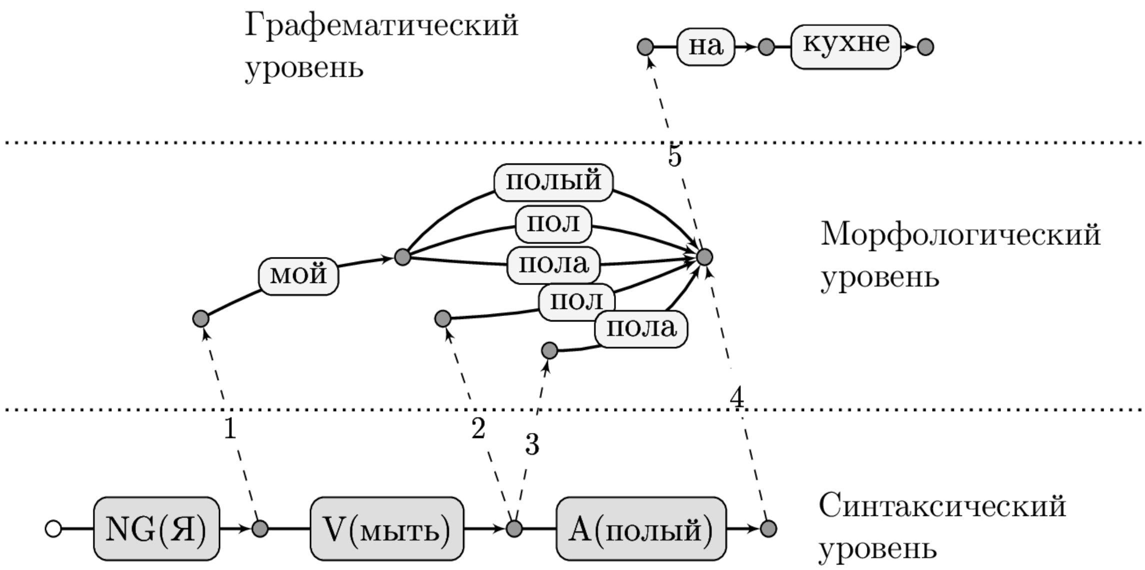


Рис. 4. Промежуточное состояние обработки графов интерпретаций

Пунктирными линиями на рис. 4 показаны и пронумерованы приостановленные процессы, стрелки указывают на входные графы. Процесс морфологического уровня (номер 5) приостановлен для словоформ *на* и *кухне*, также как и процесс синтаксического уровня (номер 1) для обработки второй интерпретации слова *мой*. Ещё три процесса уровня синтаксиса ждут соответственно обработки ещё двух интерпретаций словоформы *пол* (процессы 2 и 3) и морфологического анализа словоформы *на* (процесс 4).

Ниже на рис. 5 показано состояние графов интерпретаций на всех

рассматриваемых уровнях к моменту, когда получена первая интерпретация всей фразы *«Я мою пол на кухне»*. Видно, что при этом осталось 5 приостановленных процессов (стрелками показаны их входные графы). Построенное на синтаксическом этапе дерево разбора представлено в графе в префиксной форме: цифры в вершине-нетерминале указывают количество её дочерних вершин, в частности, для срединного узла *VG(3)* это *NG* (*Я*), *V* (мыть) и *NG* (пол).

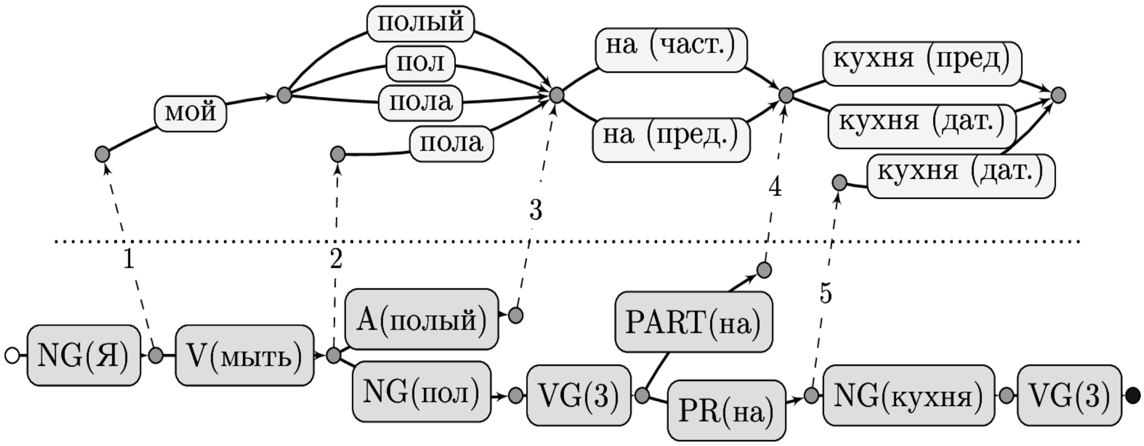


Рис. 5. Состояние анализа после получения одной интерпретации глубокого уровня

6. ФУНКЦИОНАЛЬНАЯ РЕАЛИЗАЦИЯ МОДЕЛИ

Реализация рассмотренной модели многоуровневого анализа на функциональном языке программирования (таком, как Scala) позволяет обойтись без вспомогательных структур данных для хранения состояния анализа. Модель непосредственно реализуется путём определения структур, реализующих графы интерпретаций и перевода рассмотренных выше формул-условий (1)–(3) на язык программирования.

В проведённой реализации на языке Scala для представления графов интерпретаций используется алгебраический тип данных *InterprGraph*, имеющий три альтернативы, соответствующие описанным выше видам графов:

- *EpsGraph* – тривиальный граф из одной вершины G_ε;
- *ConsGraph (firstElement, restGraph)* – граф, полученный присоединением элемента *firstElement* к графу *restGraph*;
- *UnionGraph(graph1, graph2)* – граф, полученный объединением двух графов.

Процессы анализа реализованы в виде функций, осуществляющих применение одного шага процесса анализа на основе декомпозиции входного графа интерпретаций в соответствии с видом графа (одной из

альтернатив приведённого определения). При этом после выполнения шага процесса обработанный фрагмент графа интерпретаций предыдущего уровня больше не требуется и может быть удалён сборщиком мусора.

Альтернатива *ConsGraph* является ленивой по второму аргументу *restGraph*, то есть следующий за ребром граф не вычисляется до тех пор, пока не будет запрошен каким-либо процессом анализа. Ленивость r*estGraph* позволяет хранить приостановленный процесс анализа, который может быть продолжен при необходимости вычисления этого значения.

Информация о состоянии анализа представляется набором связанных структур данных одного типа – графов интерпретаций. Множество интерпретаций может быть получено из графа в виде ленивого списка, который позволяет запрашивать одну или несколько первых интерпретаций без вычисления остальных.

Для рассматриваемой модели анализа были разработаны средства на языке программирования Scala, которые позволяют производить описание процедур анализа для каждого этапа, их применение и визуализацию результатов анализа. Описание процедур анализа осуществляется с помощью встроенного в Scala языка предметной области, включающего разноуровневые абстракции, в том числе различные виды автоматов и преобразователей, регулярные и контекстно-свободные грамматики, расширенные возможностями дополнительных проверок и хранения состояния.

7. ЗАКЛЮЧЕНИЕ

В работе рассмотрена разработанная модель многоуровневого анализа текста на естественном языке, позволяющая единообразно и естественным образом, без введения дополнительных уровней и этапов, обрабатывать множество интерпретаций текстовых единиц на разных уровнях (этапах) анализа. Ключевыми моментами являются одновременное проведение анализа на нескольких уровнях, управление (на каждом уровне) предшествующим этапом анализа с получением от него необходимых интерпретаций, возможность продолжить анализ в случае некорректности полученной интерпретации. В результате реализуемого на этой основе поиска вглубь в пространстве интерпретаций обработка фрагмента текста даёт некоторую его интерпретацию наиболее глубокого уровня, не дожидаясь полного его анализа.

Модель многоуровневого анализа текста даёт возможность:

- проводить анализ текста без его полной загрузки в память, что

позволяет обрабатывать тексты большого размера в ограниченной памяти;

- сохранить преимущества от разделения процесса анализа на независимые этапы-процедуры, которые можно разрабатывать и тестировать по отдельности;
- сделать анализ текста более эффективным за счёт распараллеливания многоуровневой обработки процессами.

Рассмотренная модель может быть дополнена различными эвристиками. Так, возможна стратегия выбора очередного приостановленного процесса анализа, в соответствии с некоторой мерой перспективности (аналогично эвристическому поиску), что может обеспечить наиболее быстрый поиск интерпретации текста при условии хорошего подбора меры.

Выбор функционального языка программирования существенно упрощает реализацию модели, позволяя обойтись без многих вспомогательных структур для хранения состояния анализа.

К прикладным задачам обработки текста на естественном языке, в которых целесообразно применение рассмотренной модели анализа, относятся задачи, требующие достаточно глубокого анализа текста и рассмотрения нескольких его интерпретаций. К таким задачам можно отнести машинный перевод текстов, извлечение информации из отдельного взятого документа, составление аннотаций текста, обработка ЕЯ-запросов в вопросно-ответных системах.

СПИСОК ЛИТЕРАТУРЫ

1. Леонтьева, Н. Н. (2006), *Автоматическое понимание текстов: Системы, модели, ресурсы*, М., Академия, 304 с.
2. Большакова, Е. И., Носков, А. А. (2010), «Анализ текста на основе лексико-синтаксических шаблонов с сокращением многовариантности», *Новые информационные технологии в автоматизированных системах: материалы тринадцатого научно-практического семинара*, М., МИЭМ, сс. 62–69.
3. Ferrucci, D., Lally, A. (2004), “UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment”, In *Natural Language Engineering*, Vol. 10, No. 3–4, pp. 327–348.
4. McKelvie, D., Brew, C., Thompson, H. (1998), Using SGML as a Basis for Data-Intensive Natural Language Proceedings, *Computers and the Humanities*, Vol. 31, No. 3–4, pp. 367–388.
5. Stoness, S. C. (2001), *Continuous Understanding: A First Look at CAFE*, University of Rochester, Department of Computer Science,

310 p.
6. Tomita, M. (1987), “An Efficient Augmented-Context-Free Parsing Algorithm”, *Computational Linguistics*, No. 13, pp. 31–46.
7. Odersky, M., et al. (2004), *An Overview of the Scala Programming Language*, EPFL Technical Report IC/2004/64, Lausanne, Switzerland, 20 p.
8. Erwig, M. (1997), “Functional Programming with Graphs”, *2nd ACM SIGPLAN International Conf. on Functional Programming*, ACM Press, pp. 52–65.
9. Frost, R. A. (2006), “Realization of Natural Language Interfaces Using Lazy Functional Programming”, *ACM Comp. Surv*ey, Vol. 38, No. 4, Article 11.
10. Hafiz, R., Frost, R. A. (2011), “Modular Natural Language Processing Using Declarative Attribute Grammars”, *Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence*, No. 7094, Berlin, Springer, pp. 291–304.
11. Gibbons, J. (1995), “An Initial-Algebra Approach to Directed Acyclic Graphs. In Mathematics of Programming Constructions”, *Lecture Notes in Artificial Intelligence*, No. 947, Berlin, Springer, pp. 282–303.

A MODEL OF MULTI-LEVEL TEXT ANALYSIS BASED ON DEPTH-FIRST SEARCH IN THE SPACE OF INTERPRETATIONS

E. I. Bolshakova, email: eibolshakova@gmail.com
A. A. Noskov, email: alexey.noskov@gmail.com
Moscow State Lomonosov University

Abstract. The paper proposes a model of natural language text analysis, which takes into account the generally accepted decomposition of the analysis process into the stages (tokenization, morphologic analysis, syntactic parsing, semantic and pragmatic analysis), but does not require intermediate auxiliary stages intended to reduce multiple homonymous analysis variants for text units (symbols, words, phrases, sentences). In the model, results of all analysis stages (levels) are represented uniformly, as graphs of interpretations, which enables to compactly store the numerous analysis variants. The key feature of the proposed model is simultaneous analysis on several levels; with the deeper processing stages govern previous stages: they call for an interpretation (analysis variant) of a text element and then checks its correctness on the current analysis level and construct an interpretation of this level. When the interpretation obtained from the previous level is incorrect, and respectively construction of interpretation on the current level is impossible, in order to continue the analysis, another interpretation is called from the prior level. Thus, the exhaustive exploration in the space of text units interpretations at all text levels is feasible. Since the proposed model implements the depth-first search in the space of interpretations, it makes it possible to construct the deepest level interpretation for initial text fragments, without performing a complete analysis of the text. The paper describes the basic operations with graphs of text interpretations and the required organization of analysis procedures at each level of text processing. The general scheme of multi-level text analysis are characterized, and some features of its implementation based on functional-object programming language with "lazy" computing are also pointed out.

Key words and phrases: natural language text analysis; levels and stages of text analysis; the space of text interpretations; graphs of interpretations; model of multi-level text analysis.

Computing Classification System 1998: I.2.7

Mathematics Subject Classification 2010: 68T50

REFERENCES

1. Leont'eva, N. N. (2006), Automatic understanding of texts: Systems, models, resources [Avtomaticheskoe ponimanie tekstov: Sistemy, modeli, resursy], Moscow, Akademija Publ., 304 p.
2. Bolshakova, E. I., Noskov, A. A. (2010), "Text analysis on the basis of lexical-syntactic patterns with the reducing of multivariance", *New information technologies in automated systems: Proceedings of 13th scientific seminar* ["Analiz teksta na osnove leksiko-sintaksicheskih shablonov s sokrascheniem mnogovariantnosti", *Novye informatsionnye tehnologii v avtomatizirovannyx sistemah: materialy trinadtsatogo nauchno-prakticheskogo seminara*], Moscow, MIEM, pp. 62–69.
3. Ferrucci, D., Lally, A. (2004), "UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment", In *Natural Language Engineering*, Vol. 10, No. 3–4, pp. 327–348.
4. McKelvie, D., Brew, C., Thompson, H. (1998), Using SGML as a Basis for Data-Intensive Natural Language Proceedings, *Computers and the Humanities*, Vol. 31,

No. 3–4, pp. 367–388.

5. Stoness, S. C. (2001), *Continuous Understanding: A First Look at CAFE*, University of Rochester, Department of Computer Science, 310 p.
6. Tomita, M. (1987), "An Efficient Augmented-Context-Free Parsing Algorithm", *Computational Linguistics*, No. 13, pp. 31–46.
7. Odersky, M., et al. (2004), *An Overview of the Scala Programming Language*, EPFL Technical Report IC/2004/64, Lausanne, Switzerland, 20 p.
8. Erwig, M. (1997), "Functional Programming with Graphs", *2nd ACM SIGPLAN International Conf. on Functional Programming*, ACM Press, pp. 52–65.
9. Frost, R. A. (2006), "Realization of Natural Language Interfaces Using Lazy Functional Programming", *ACM Comp. Surv*ey, Vol. 38, No. 4, Article 11.
10. Hafiz, R., Frost, R. A. (2011), "Modular Natural Language Processing Using Declarative Attribute Grammars", *Advances in Artificial Intelligence*, *Lecture Notes in Artificial Intelligence*, No. 7094, Berlin, Springer, pp. 291–304.
11. Gibbons, J. (1995), "An Initial-Algebra Approach to Directed Acyclic Graphs. In Mathematics of Programming Constructions", *Lecture Notes in Artificial Intelligence*, No. 947, Berlin, Springer, pp. 282–303.

УДК 003.26; 004.421

ПРОЗРАЧНАЯ РЕАЛИЗАЦИЯ ПОДПИСИ В СХЕМЕ EDDSA*

В. И. Лосев, email: i2porignal@yandex.ru
независимый исследователь

Аннотация. Работа посвящена вопросам реализации криптографии и системы электронной подписи EdDSA, предложенной Даниелом Бернштейном, выполненной на основе частного случая кривой Эдварда со специальным набором параметров. Данная криптография используется всё в большем числе приложений, таких как OpenSSH, Tor, I2P. Фактически же используются разновидности авторской реализации ref10, изначальной написанной на Ассемблере для amd64 и портированной на другие языки и платформы. Несмотря на её эффективность, ref10 является запутанной и сложной для понимания даже специалистами, с большим объемом исходного кода, изобилующего «магическими» константами. В работе рассматриваются математические основы вычислений на кривой в терминах операций с большими целыми числами и практическая реализация на языке Си++ с использованием популярной криптографической библиотеки OpenSSH, в отличие от ref10, использующей свою собственную реализацию. Переход к операциям из стандартной библиотеки позволяет резко уменьшить объем исходного кода, улучшить его читаемость и при необходимости использовать любую другую библиотеку. Прозрачная реализация позволяет дальнейшую оптимизацию для конкретных задач, а также может служить образовательным целям.
Операции над кривой состоят из сложения двух точек и умножения на константу. Рассмотрены методы их эффективной реализации, постепенное применение которых позволяет достичь производительности, сравнимой с ref10. Рассмотрено представление точки на кривой с помощью одной координаты Y и вычисление координаты X в поле вычетов по модулю.
Показана реализация электронной подписи EdDSA по схеме Бернштейна и использованием операций над кривой и хэш-функции SHA-512. Методы сокращения объема вычислений при проверке подписи и использование набора предварительно вычисленных точек на кривой.

Ключевые слова и фразы: эллиптическая кривая; электронная цифровая подпись; криптостойкость; программная реализация.

1 ВВЕДЕНИЕ

В последнее время всё большую популярность набирает электронная подпись Ed25519, основанная на разновидности эллиптической кривой, предложенной Бернштейном [1]. По мере увеличения числа узлов

I2P (проект «Невидимый интернет»[1], [2]) с данным видом подписи, в реализации I2P возникла необходимость её программной поддержки – поскольку подпись Ed25519 не входит в состав популярных криптографических библиотек. Как правило, при этом используются разновидности ref10 из библиотеки SUPERCOP [3], реализованной самим Бернштейном на ассемблере и затем портированной на другие языки. Данная реализация работает хорошо и быстро, однако у неё есть существенный недостаток – она непонятна. Действительно, если заглянуть в её исходный код, то можно увидеть большое количество однотипных строк, оперирующих с множеством «магических» чисел, понять же, что они означают, без углубления в теорию не представляется возможным. Целью данной статьи является математически прозрачная реализация Ed22519, использующая лишь стандартные операции с большими числами, присутствующие в любой криптографической библиотеке, причём имеющая скорость работы, достаточную для практического использования в I2P.

2 ПРИНЦИП РАБОТЫ ЭЛЛИПТИЧЕСКОЙ КРИПТОГРАФИИ – ОСНОВНЫЕ СВЕДЕНИЯ

Задаётся неявное уравнение плоской кривой специального вида, называемого эллиптической кривой [4]. Также задаётся простое число, образующее поле вычетов по модулю. Координаты точек кривой принадлежат этому полю, т.е. являются целыми неотрицательными числами, не превышающее модуль. Над двумя точками кривой задаётся операция сложения – таким образом, чтобы новая точка также принадлежала кривой. Если точку сложить саму с собой, то получится умножение на 2, если прибавить ещё раз – то на 3, и так далее; т.е. таким образом определяется умножение на произвольное число n. Также вместе с кривой задаётся базовая точка, принадлежащая кривой, операции над которой и используются в криптографии.

Для криптографии выбирается произвольное (достаточно большое) число n, являющееся секретным ключом; на него умножается базовая точка, и новая точка служит публичным ключом – который затем умножается противоположной стороной на другое число с целью согласования ключа или проверки подписи. Стойкость ключа основывается отсутствием известного на данный момент способа *определения множителя по точке* за разумное время.

---

[1] Терминология не устоялась, поэтому мы предлагаем писать это название *со строчной буквы* – в отличие от единственного в своём роде «большого» Интернета. *(Прим. ред.)*

3 СХЕМА Ed25519

Берштейн предложил использовать эллиптическую кривую со следующими параметрами.

- Уравнение кривой:

$y^2 - x^2 = 1 + d \cdot x^2 \cdot y^2$, где $d = \frac{-121665}{121666}$.

- Модуль:

$q = 2^{255} - 19$ (отсюда и возникло название кривой).

- Базовая точка:

(x, 4/5), где x получается решением приведённого далее уравнения. Её координаты в явном виде:

x=15112221349535400772501151409588531511454012693041857206046113283949847762202,
y=46316835694926478169428394003475163141307993866256225615783033603165251855960.

- Сложение:

$(\frac{x1 \cdot y2 + x2 \cdot y1}{1+m}, \frac{x1 \cdot x2 + y1 \cdot y2}{1-m})$, где $m = d \cdot x1 \cdot x2 \cdot y1 \cdot y2$.

Следует заметить, что всюду в рассматриваемых нами формулах деление – это не обычное деление с остатком, а умножение на обратный по модулю элемент, вычисляемый согласно малой теоремы Ферма [5] по формуле x^{q-2}, т.е. используется операция возведения в степень по модулю.

В качестве публичного ключа при этом используется только координату y, поэтому при необходимости решается уравнение кривой относительно *x*; применяется формула $\sqrt{\frac{y^2-1}{d \cdot y^2+1}}$. Поскольку q представимо в виде $8*k+5$, для вычисления квадратного корня из x следует воспользоваться формулой x^{k+1}. Действительно,

$$q = 2^{255} - 24 + 5 = 8 * (2^{252} - 3) + 5 ,$$

отсюда квадратный корень равен $x^{2^{252}-2}$.

4 РЕАЛИЗАЦИЯ АЛГОРИТМА

Код полностью располагается по адресу [6] в файле Signature.cpp. Для работы с большими числами используются функции библиотеки BN из OpenSSH [7].

Создадим класс Ed25519, реализующий саму кривую и содержащий рассчитываемые параметры кривой. В первую очередь необходимы 3 метода: сложение, умножение на число и вычисление x по заданному y.

```
class Ed25519
{
```

```
//...
    EDDSAPoint Sum (const EDDSAPoint& p1, const EDDSAPoint& p2) const
    EDDSAPoint Mul (const EDDSAPoint& p, const BIGNUM * e) const
    BIGNUM * RecoverX (const BIGNUM * y) const
//...
}
```

Из-за частного использования операция сложения и трудоёмкости операции деления приведём *x* и *y* к общему знаменателю *(1+m)·(1-m)*, тем самым избавившись от лишней операции деления. В результате код для сложения выглядит следующим образом:

```
// m = d*p1.x*p2.x*p1.y*p2.y
BN_mul (xx, p1.x, p2.x, m_Ctx); //p1.x*p2.x
BN_mul (yy, p1.y, p2.y, m_Ctx);  //p1.y*p2.y
BIGNUM * m = BN_dup (d);
BN_mul (m, m, xx, m_Ctx);
BN_mul (m, m, yy, m_Ctx);
// x = (p1.x*p2.y + p2.x*p1.y)*inv(1 + m)
// y = (p1.y*p2.y + p1.x*p2.x)*inv(1 - m)
// m1 = 1-m
BN_one (m1);
BN_sub (m1, m1, m);
// m = m+1
BN_add_word (m, 1);
// y = (p1.y*p2.y + p1.x*p2.x)*m
BN_add (y, xx, yy);
BN_mod_mul (y, y, m, q, m_Ctx);
// x = (p1.x*p2.y + p2.x*p1.y)*m1
BN_mul (yy, p1.x, p2.y, m_Ctx);
BN_mul (xx, p2.x, p1.y, m_Ctx);
BN_add (x, xx, yy);
// denominator m = m*m1
BN_mod_mul (m, m, m1, q, m_Ctx);
Inv (m);
BN_mod_mul (x, x, m, q, m_Ctx); // x = x/m
BN_mod_mul (y, y, m, q, m_Ctx); // y = y/m
```

Для удвоения (сложения точки с собой) реализован отдельный метод `Double`, поскольку в этом случае `p1.x = p2.x` и `p1.y = p2.y`, что позволяет сократить число умножений. Кроме того, вместо `BN_mul` используется более быстрая `BN_sqr`.

Умножение реализовано простейшим методом удвоения и прибавления, т.е. необходимо пройти вдоль числа побитово (от старшего бита к младшему), на каждом шаге удваивать значение результата, а если бит – единица, то прибавлять умножаемую точку.

```
EDDSAPoint res {zero, one};
if (!BN_is_zero (e))
{
    int bitCount = BN_num_bits (e);
    for (int i = bitCount - 1; i >= 0; i--)
    {
        res = Double (res);
        if (BN_is_bit_set (e, i)) res = Sum (res, p);
    }
}
```

Вычисление x по y реализовано тривиально. Сначала подкоренное выражение:

```
BN_sqr (y2, y, m_Ctx); // y^2
// xx = (y^2 -1)*inv(d*y^2 +1)
BN_mul (xx, d, y2, m_Ctx);
BN_add_word (xx, 1);
Inv (xx);
BN_sub_word (y2, 1);
BN_mul (xx, y2, xx, m_Ctx);
```

Затем вычисление квадратного корня по формуле $x^{2^{252}-2}$:

```
// x = srqt(xx) = xx^(2^252-2)
BN_mod_exp (x, xx, two_252_2, q, m_Ctx);
```

5 ПОДПИСЬ И ПРОВЕРКА ПОДПИСИ

Данная тема достаточно интересна и обширна, потому автор предполагает посвятить этому вопросу отдельную статью. В то же время, методы `Sign` и `Verify` реализованы и используются в практических приложениях. Потому здесь будут перечислены лишь некоторые особенности реализации.

Как и другие системы электронной подписи, подпись EdDSA представляет собой пару 32-байтных чисел (R,S), потому длина подписи – 64 байта. Числа представлены в Little Endian [8]. В качестве хэш-функции используется SHA-512 [9]. При подписи случайное число не генерируется, вместо этого используется правая половина SHA-512 хэша от секретного ключа, объединенного с самими подписываемыми данными. Также используется простое число

$$2^{252} + 27742317777372353535851937790883648493 ,$$

ранее уже использовавшееся при выборе базовой точки – с тем расчётом, чтобы умножение на него базовой точки давало нуль.

6 ЗАКЛЮЧЕНИЕ

Очевидно, что самым медленным местом данной реализации является деление в операции сложения. На самом деле, пользуясь современными достижениями теории чисел и спецификой параметров EdDSA, можно исключить его полностью, как это сделано в ref10 [3]. Однако это существенно усложнит реализацию и сделает её менее понятной, поэтому такие изменения программы следует делать только в случае возникновения реальной практической необходимости. В настоящее время проверка подписи EdDSA в I2P – довольно редкое событие (по сравнению, скажем, с шифрованием по схеме Эль-Гамаля [10]).

Существует мнение, что реализация собственной криптографии – крайне плохая идея. Однако использование «непонятно как работающей

реализации» представляется немногим лучше. Кроме того, данная статья должна быть полезна тому, кому интересно докопаться до сути, а работающий практический код в этом поможет.

СПИСОК ЛИТЕРАТУРЫ

1. Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y. (2012), "High-speed high-security signatures", *Journal of Cryptographic Engineering*, Vol. 2, No. 2, pp. 77–89. doi: 10.1007/s13389-012-0027-1.
2. «I2P: Официальный сайт проекта» (2015), режим доступа: https://geti2p.net/ru/about/intro
3. "eBACS: ECRYPT Benchmarking of Cryptographic Systems" (2015), режим доступа: http://bench.cr.yp.to/supercop.html
4. Острик, В. В., Цфасман, М. А. (2001), *Алгебраическая геометрия и теория чисел: рациональные и эллиптические кривые*, МЦНМО, М., 48 с.
5. Боревич, З. И., Шафаревич, И. Р. (1972), *Теория чисел*, Наука, М., 175 с.
6. "Code, Manage, Collaborate" (2015), страница "orignal", режим доступа: https://bitbucket.org/orignal/i2pd/src
7. "OpenSSL. Cryptography and SSL/TLS Toolkit" (2015), режим доступа: https://www.openssl.org/docs/manmaster/crypto/bn.html
8. Таненбаум, Э. (2007), *Архитектура компьютера*, Питер, СПб., 844 с.
9. Фергюсон, Н., Шнайер, Б. (2004), *Практическая криптография*, Диалектика, М., 432 с.
10. Menezes, A. J., van Oorschot, P. C., Vanstone, S. A. (1996), *Applied Cryptography*, CRC Press, 816 p.

A TRANSPARENT IMPLEMENTATION OF THE SIGNATURE IN SCHEME EDDSA

V. I. Losev, email: i2porignal@yandex.ru
independent researcher

Abstract. The paper describes detailed implementation of digital signature system EdDSA. This is a special case of the Edward twisted curves family, introduced by Daniel J. Bernstein (aka DJB) from University of Illinois (Chicago), with special choice of the basic parameters.

This crypto becomes more and more popular and used by strong crypto applications like OpenSSH, Tor, I2P. However, all of these applications uses DJB's implementation ref10, written by amd64 assembly language and ported to many other languages and platforms. Despite unbeatable performance, the code of his implementation seems overcomplicated and not easy to understand even by a specialist, with huge amount of the source code and bunch of "magic" numbers in it. This paper is trying to sort this things out from curve's math expressed in operations over big number to real C++ code doing it with popular crypto library OpenSSH, unlike ref10 with own implementation of big numbers.

Using generic big numbers reduces an amount of the source code, improves readability, and opens a way to use any other crypto library in future. The same time, the transparent implementation leaves a room for further optimization for particular tasks and might serve to educational purposes.

There are two major operation over the curve: addition of two point and multiplication by constant. This paper considers few ways of their efficient implementation, applying them step-by-step let you reach a good performance, comparable with ref10. A point can be presented by its Y-coordinate only; a recovery method of X-coordinate from Y in a finite field is described.

The complete EdDSA digital signature system uses DJB's schema with operations over the curve and SHA-512 as hash functions. Few approaches to reduce the amount of calculation during signature verification are shown. Also possible pre-calculations to speed verification up.

Keywords and phrases: elliptic curve; electronic digital signature; cryptographic; software implementation.

Computing Classification System 1998: E.3

Mathematics Subject Classification 2010: 14G50

REFERENCES

1. Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y. (2012), "High-speed high-security signatures", *Journal of Cryptographic Engineering*, Vol. 2, No. 2, pp. 77–89. doi: 10.1007/s13389-012-0027-1.
2. "I2P: The Invisible Internet Project" (2015), access mode: https://geti2p.net/ru/about/intro
3. "eBACS: ECRYPT Benchmarking of Cryptographic Systems" (2015), access mode: http://bench.cr.yp.to/supercop.html
4. Ostrik, V. V., Cfasman, M. A. (2001), *Algebraic geometry and number theory: rational and elliptic curves [Algebraicheskaya geometriya i teoriya chisel: racional'nye i ellipticheskie krivye]*, MCNMO Publ., Moscow, 48 p.
5. Borevich, Z. I., Shafarevich, I. R. (1972), *Number theory [Teoriya chisel]*, Nauka, Moscow, 175 p.

6. “Code, Manage, Collaborate” (2015), page “orignal”, access mode: https://bitbucket.org/orignal/i2pd/src
7. “OpenSSL. Cryptography and SSL/TLS Toolkit” (2015), access mode: https://www.openssl.org/docs/manmaster/crypto/bn.html
8. Tanenbaum E. (2007), *Computer architecture [Arhitektura komp’yutera]*, Piter Publ., SPb, 844 p.
9. Ferguson N., Schneier, B. (2003), *Practical Cryptography: Designing and Implementing Secure Cryptographic Systems*, Wiley Publ., 400 p.
10. Menezes, A. J., van Oorschot, P. C., Vanstone, S. A. (1996), *Applied Cryptography*, CRC Press, 816 p.

УДК 519.683.8

ПОДХОД К РЕАЛИЗАЦИИ МЕТОДА ВЕТВЕЙ И ГРАНИЦ ДЛЯ ЗАДАЧ ДИСКРЕТНОЙ ОПТИМИЗАЦИИ. ЧАСТЬ I *

Б. Ф. Мельников, email: bf-melnikov@yandex.ru
Е. А. Мельникова, email: ya.e.melnikova@yandex.ru
Самарский государственный университет

Аннотация. Главная цель статьи – рассмотрение легко реализуемого метода решения целого класса переборных задач. Его можно назвать подходом к реализации комплекса алгоритмов, построенных на основе метода ветвей и границ, включающего многие дополнительные однотипные эвристики; эта однотипность заключается в том, что сами эвристики практически не изменяются при переходе от одной задачи к другой. Поэтому мы называем этот метод мультиэвристическим подходом. Нередко при этом получается комплекс эвристических алгоритмов для практического решения некоторой NP-трудной задачи.

По мнению авторов, во многих научных публикациях, посвящённых описаниям алгоритмов, а также в различных учебниках, на саму реализацию программ нередко «накладывается табу». Более того, нередко встречается худшая ситуация: сами тексты программ, приведённые в очень хороших книгах, кажутся несколько «недоделанными» – потому что авторы практически не уделяют внимания «чисто программистским» вопросам.

А второй целью статьи является рассмотрение конкретного примера одной из таких задач – причём это рассмотрение включает азы необходимой для её решения математической теории. В статье нами приведено элементарное изложение теории минимизации недетерминированных конечных автоматов – той её части, которая нужна для читателей-программистов, не имеющих необходимости рассматривать эту область подробно.

Ключевые слова и фразы: задача дискретной оптимизации; эвристический алгоритм; программная реализация; метод ветвей и границ.

1 ВВЕДЕНИЕ

Основные цели настоящей статьи таковы. Во-первых, главная цель – мы приводим легко реализуемый *метод* решения целого класса переборных задач. Его можно назвать подходом к реализации комплекса алгоритмов, построенных на основе метода ветвей и границ, включающего многие дополнительные *однотипные* эвристики; эта однотипность заключается в том, что сами эвристики практически не изменяются при переходе от одной задачи к другой. Поэтому, согласно [1–4] и др., мы назвали этот метод *мультиэвристическим подходом*. Слож-

ность задач, решаемых нами с помощью этого подхода, весьма различна – от «студенческих олимпиадных» задач до больших программных проектов, нередко представляющих собой комплекс эвристических алгоритмов для *практического* решения некоторой NP-трудной задачи. Однако в наших предыдущих публикациях *о реализации* алгоритмов практически ничего сказано не было – и в данной статье мы этот недостаток устраняем.

По мнению авторов, во многих научных публикациях, посвящённых описаниям алгоритмов, а также в различных учебниках, на саму реализацию программ нередко «накладывается табу». Более того, нередко встречается и гораздо более худшая ситуация – следующая. Сами тексты программ, приведённые в очень хороших книгах (вроде [5, 6]), кажутся несколько «недоделанными» – просто потому, что авторы практически не уделяют внимания «чисто программистским» вопросам; среди этих вопросов мы в первую очередь отмечаем выделение и освобождение динамической памяти и т. п. – но не только их. Причём для реализации описываемых нами программ, по-видимому, вполне достаточно знание малого подмножества Си++ – малого, но «хорошего» подмножества, *самодостаточного*.

А второй целью статьи является рассмотрение конкретного примера одной из таких задач – причём это рассмотрение включает азы необходимой для её решения математической теории. Итак, нами приведено очень краткое (элементарное) изложение теории минимизации недетерминированных конечных автоматов [7, 8] – той её части, которая нужна, например, для читателей-программистов, не имеющих необходимости рассматривать эту область подробно.

Перед изложением основных алгоритмов мы приводим полное решение более простой переборной задачи, выполненное с помощью того же самого подхода.

Ещё отметим, что данная статья фактически является переработкой авторского материала, являющегося специальным курсом для магистрантов программистских специальностей.

2 ОДНА НЕСЛОЖНАЯ ЗАДАЧА

Начнём с одной из задач финала ACM 2007 года – а именно, с задачи F, см. [9, 10]. Это – задача финала самого престижного международного студенческого чемпионата мира (пусть даже одна из наиболее простых предлагавшихся там задач), однако её уровень примерно таков, что при решении «в режиме off-line» её сложность вполне доступна студенту-старшекурснику. Однако, как уже было отмечено выше, при её рассмотрении мы фактически получаем *метод*

решения подобных переборных задач.

Итак, приведём условие задачи. (И заранее отметим, что мы немного упросили формулировку условия – например, будем рассматривать только случай $N=4$ – а также не заменяли в формулировке слова «синий цвет»: даже в случае чёрно-белого рисунка смысл очевиден.)

Для игры Marble с M шариками используется квадратная доска, разделённая на N×N клеток; из них M клеток содержат лунки. Шарики и лунки перенумерованы от 1 до M. Цель игры Marble состоит в том, чтобы положить каждый шарик в лунку со своим номером.

Доска может содержать стенки. Каждая стенка имеет длину в одну из N×N частей и разделяет две смежные клетки – т. е. клетки, имеющие общую сторону.

В начале игры все шарики расположены на доске, каждый в своей клетке. Ход состоит в том, что мы приподнимаем одну из сторон доски. При этом каждый шарик катится в направлении противоположной стороны доски, пока не столкнётся со стенкой, или не упадет в лунку, или не дойдёт до такой позиции, что следующая клетка занята другим шариком. Итак, движение шарика ограничено следующими условиями.

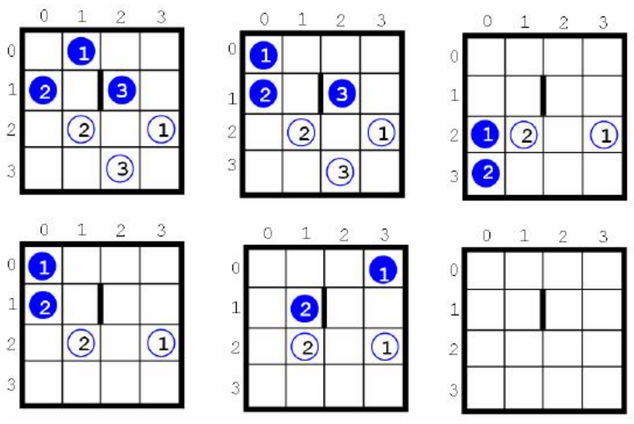


Рис. 1

- *Шарик не может перескочить через стенку, другой шарик или лунку.*
- *Шарик не может покинуть доску (край доски ограничен стенкой).*
- *В клетке не могут находиться более одного шарика в одно и то же время.*
- *Если шарик попадает в клетку с лункой, то лунка ста-*

новится заполненной, и другие шарики могут перекатываться через эту клетку (а также останавливаться в ней). Шарик, уже оказавшийся в лунке, покинуть её не может.

Игра заканчивается успешно, если каждый шарик находится в лунке со своим номером.

На рис.1 приведено решение игры для доски 4×4 с 3 синими шариками, 3 лунками и одной стенкой. (Минимальное) решение состоит из 5 ходов: поднять восточную сторону, затем северную, далее южную, западную, и снова северную.

Программа должна определить минимально возможное число ходов для решения задачи – если решение вообще существует; иначе нужно выдать, что решения нет.

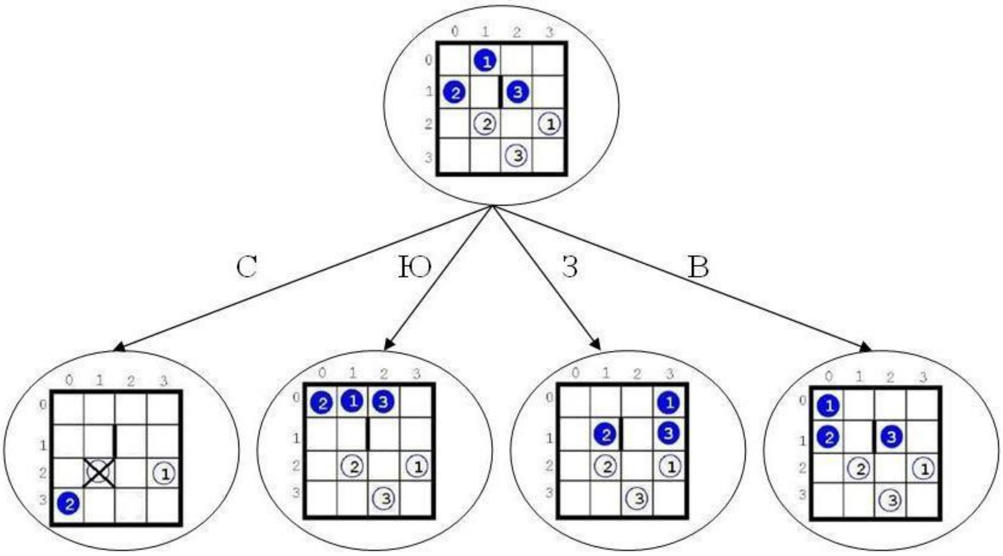


Рис. 2

Перед обсуждением решения сто́ит отметить, что вместо термина «игра» точнее употреблять слово «головоломка» – что больше соответствует терминологии, применяемой в литературе по искусственному интеллекту ([11, 12] и др.; также см. [13, разд. 5.5]); однако мы при переводе условия задачи оставили терминологию оригинала. А раз это головоломка – то можно применить обычный метод их решения, т.е. полный перебор в пространстве состояний. Но, как уже было отмечено во введении, сам наш подход к решению подобных задач, к организации перебора в них (точнее – перебора с возвратами, бэктрекинга) может быть применён и при решении других, значительно более сложных проблем. Мы на рис. 2 приводим только «начало» дерева перебора (каждая вершина которого является позицией) – его куст, относящийся к корню (исходной позиции).

Итак, полный перебор; при этом мы должны, во-первых, предотвращать возможные зацикливания, во-вторых, находить оптимальное решение (минимальное по числу ходов), и, в-третьих, уметь находить ситуации, в которых решения нет. Всё это приводит к мысли использовать поиск в ширину – что мы и сделаем.

Мы не будем сохранять всё дерево перебора. Здесь – а ещё важнее, что *всюду в подобных ситуациях* – достаточно (и желательно!) хранить только листья этого дерева; а раз мы решили организовать поиск в ширину – то листья желательно хранить в виде очереди, см. [14] и др. (А в случае поиска в глубину самой удобной структурой данных, по-видимому, является стек, который тоже удобно организовывать на основе массива или списка.) Мы реализуем очередь с помощью самого простого массива (указателей на вершины-позиции). В приведённой далее программе каждая такая вершина будет иметь тип `MyBoard`. А предупреждать возможное зацикливание мы будем самым простым возможным способом – будем запоминать все встретившиеся позиции.[1]

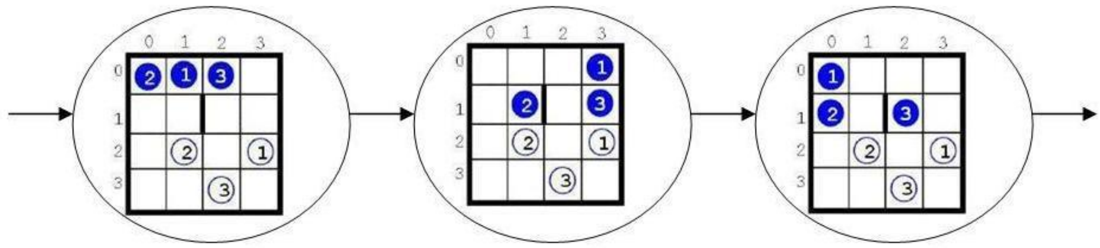


Рис. 3

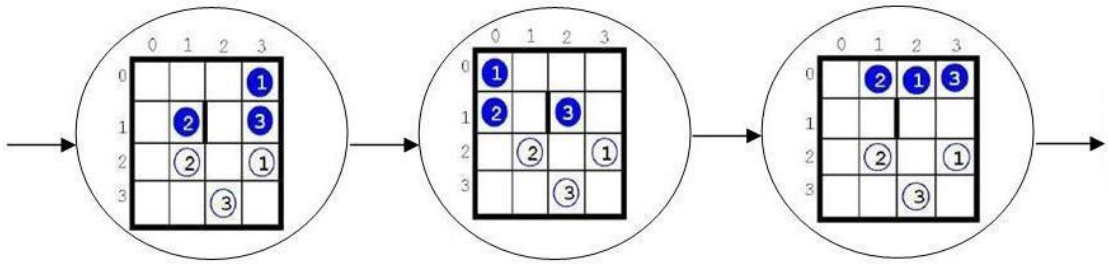


Рис. 4

Итак, у нас в программе будет даже 2 массива – в одном мы будем хранить ещё не рассмотренные позиции, а в другом – уже рассмотренные. В начале работы первый из них (главный) состоит из единственного элемента (соответствующего стартовой позиции) – второй же вообще пуст. Далее мы последовательно удаляем первый

[1] Заранее отметим, что в более сложных задачах этот путь, конечно же, неприемлем – и нужно искать другие выходы.

элемент главного массива, текущую позицию (добавляя её одновременно во второй массив) – и вместо неё записываем *в конец* главного массива (очередь!) *три* позиции, каждая из которых может быть получена из текущей за 1 ход[2]. Значения главного массива (списка) после первой и второй итераций приведены соответственно на рисунках 3 и 4.

Перейдём к описанию программной реализации. Мы используем объектно-ориентированное программирование и простейшие классы библиотеки MFC [15]. Итак, возможно следующее описание класса-позиции (`MyBoard`):

```
class MyBoard {
private:
    int nDim;  // размерность
    int Board [4][4];
    bool WallRight [4][3]; // стена справа от этой клетки
    bool WallDown  [3][4]; // стена снизу от этой клетки
    int nDeep;
        // уровень, глубина перебора
        // (число уже сделанных ходов)
    int nLast;
        // последний сделанный ход:
        // 0 – не знаем, 1 – север, и т.д.
public:
    MyBoard ();
    MyBoard (MyBoard& copy);  // конструктор копии
    int GetDim () { return nDim; }
    int Get (int i, int j) { return Board[i][j]; }
    int GetDeep () { return nDeep; }
    int GetLast () { return nLast; }
    friend istream& operator>>
        (istream& is, MyBoard& board);
    friend ostream& operator<<
        (ostream& os, MyBoard& board);
    bool Empty ();
        // всё ли УЖЕ хорошо? пуста ли уже доска?
    bool North ();
    // поднять доску с северной стороны;
    // возвращает true,
```

---

[2] Точнее – *не более трёх* позиций. Причём не 4 позиции – поскольку один и тот же ход бессмысленно делать два раза подряд; кстати, именно для этой цели мы ниже в классе `MyBoard` используем поле `nLast`. Единственное возможное исключение – 4 позиции могут быть добавлены в первый раз, т.е. при обработке корня дерева перебора. И, конечно, мы не записываем в массивы ни «плохие» позиции (аналогичные левой нижней на рис. 2), ни встретившиеся повторно.

```
    // если ни один из шариков не попал в чужую лунку;
    // иначе false
    bool South (); // поднять доску с южной стороны
    bool West ();  // поднять доску с западной стороны
    bool East ();  // поднять доску с восточной стороны
};
bool operator== (MyBoard& board1, MyBoard& board2);
    // сравнивает только массивы Board
    // рассматриваемых объектов
```

Некоторые комментарии уже были приведены в само́м описании класса. Реализация методов, по-видимому, вряд ли вызовет интерес – здесь это чисто техническая работа.

Для хранения массивов позиций (точнее, указателей на позиции) удобно использовать специально написанный наследник класса `CPtrArray`; важно отметить, что такой *подход* желателен очень часто – при рассмотрении массива (или списка) указателей на самые разные структуры данных. Итак, возможно следующее описание наследника класса `CPtrArray`.[3]

```
class MyArray : public CPtrArray {  // of MyBoard*
public:
    bool Exists (MyBoard* pBoard);
    void MyAdd (MyBoard* pBoard);
};
```

В отличие от рассмотренного ранее класса `MyBoard`, здесь мы приведём реализацию этих методов.

```
bool MyArray::Exists (MyBoard* pBoard) {
    for (int i=0; i<GetSize(); i++) {
        MyBoard* pN = (MyBoard*)GetAt(i);
        if (*pN==*pBoard) return true;
    }
    return false;
}
void MyArray::MyAdd (MyBoard* pBoard) {
    if (Exists(pBoard)) delete pBoard; else Add(pBoard);
}
```

---

[3] И уже сейчас, при рассмотрении не очень сложной задачи, сто́ит отметить следующее обстоятельство. Массив (или список) позиций (а в более сложных ситуациях – *подзадач*) можно упорядочивать по разным критериям – а не только по расстоянию от корня в дереве поиска, как в нашем случае, т.е. при поиске в ширину. Даже в нашем примере мы могли бы применять и другие критерии упорядочивания – например, по возрастанию числа оставшихся шариков, что как-то характеризует «близость позиции к решению». Отметим ещё, что подобное упорядочивание можно связать с решением ещё одной задачи из [8] – а именно, проблемы B.

Здесь метод `Exists()` путём простейшего линейного поиска даёт ответ на вопрос о наличии в массиве некоторого элемента-позиции (для возможности такого поиска мы ранее перегрузили оператор сравнения класса `MyBoard`) а метод `MyAdd()` добавляет в конец массива новую позицию. Если рассматриваемая позиция в массиве уже имеется, то мы не добавляем её; более того, вызываем для неё деструктор. Заметим, что подобный подход требует аккуратности (поскольку заведение объекта происходит на одном уровне работы программы, а его удаление – на другом) – однако вполне возможен.

Теперь рассмотрим наиболее важный класс `MyTask` – определяющий нашу главную конструкцию, всю задачу. Очень важно отметить, что практически такой же класс желательно применять и в более сложных задачах дискретной оптимизации – и не только в рассмотренной далее задаче минимизации автоматов, но и в ещё более сложных.

```
class MyTask {
private:
  MyArray Current;  // ещё не рассмотренные позиции
  MyArray Old;      // уже рассмотренные позиции
public:
  MyTask (MyBoard& board);
  friend ostream& operator<< (ostream& os, MyTask& task);
  int Step ();
    // возвращает:
    // -1 - массив подзадач пуст;
    // 0 - требуется продолжение работы;
    // >0 - ответ
  int Run ();
    // возвращает:
    // -1 - нет решения;
    // >=0 - ответ
};
```

Метод `Run()` не представляет интереса – он заключается в последовательных вызовах метода `Step()`, реализация которого и приведена ниже.

```
int MyTask::Step () {
  if (this->Current.GetSize()<=0) return -1;
  MyBoard* pOld = (MyBoard*)Current.GetAt(0);
  if (pOld->Empty()) return pOld->GetDeep();
      // на всякий случай перестрахуемся
  this->Current.RemoveAt(0);
  int nLast = pOld->GetLast();
  if (nLast!=1) { // можно поднять доску с северной стороны
    MyBoard* pN = new MyBoard(*pOld);
```

```
    if (!pN->North()) delete pN;
    else if (pN->Empty()) {
        int n = pN->GetDeep();
        delete pN;
        return n;
    }
    else if (Old.Exists(pN)) delete pN;
    else this->Current.MyAdd(pN);
  }
  // ...
  // аналогичные вызовы подъёма доски с 3 других сторон –
  // текст этих вызовов опущен
  // ...
  Old.Add(pOld);
  return 0;
}
```

Заметим, что можно упростить полный перебор (т.е. сократить число рассматриваемых позиций) разными способами – и не только *не* рассмотрением одного и того же хода два раза подряд (т.е. использованием поля `nLast`). Однако мы не будем подробнее обсуждать этот момент – мы надеемся вернуться к нему в статье-продолжении.

3. ЭЛЕМЕНТАРНЫЕ СВЕДЕНИЯ О МИНИМИЗАЦИИ НЕДЕТЕРМИНИРОВАННЫХ КОНЕЧНЫХ АВТОМАТОВ

Даже для не очень полного введения в задачу минимизации недетерминированных конечных автоматов (НКА), поставленную ещё, например, в книге [16], нужен значительно больший объём текста, чем может вместить данная статья. Некоторые вопросы минимизации НКА, описывающие подход авторов, приведены в работах [3, 7, 8] и др. [4]; некоторые альтернативные подходы можно увидеть, например, в [18, 19, 20] и по ссылкам, приведённым в этих работах. Однако мы надеемся, что приведённый нами в данном разделе текст вполне достаточен для почти полного понимания как рассматриваемой нами далее теории, так и приведённых далее, в части II, алгоритмов и программ на Си++. Итак, рассмотрим недетерминированные конечные автоматы – причём т.н. автоматы Рабина-Скотта (Медведева).

К классической теории, изложенной, например, в [16, 21], очень важно добавить следующее замечание. Для каждого автомата можно легко построить т.н. зеркальный (изменив на противоположные направления всех его стрелок – дуг, входов и выходов) – который за-

---

[4] В [8] мы назвали этот подход «подходом Полака» – следуя, например, работе последнего [17].

даёт зеркальный язык (где каждое слово исходного языка читается справа налево).

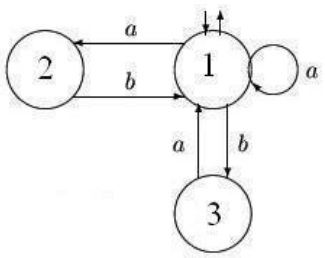


Рис. 5

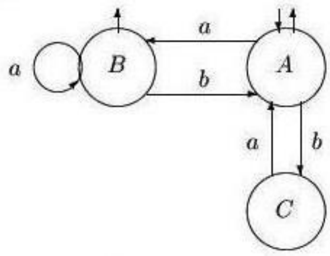


Рис. 6

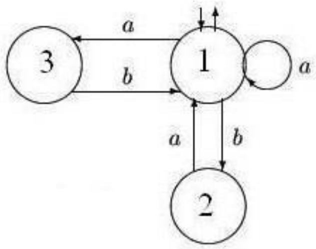


Рис. 7

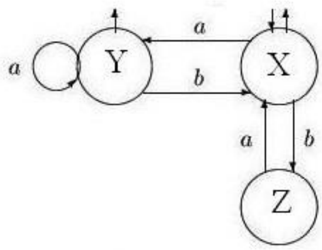


Рис. 8

Рассмотрим простой пример. Пусть исходный автомат изображён на рис. 5, а эквивалентный ему канонический – на рис. 6. Отметим, что при этом в процессе детерминизации определяется т.н. соответствие вершин: вершине 1 соответствуют две вершины – A и B[5], вершине 2 – только B, а вершине 3 – только C.

Теперь рассмотрим зеркальный (для исходного) автомат – он приведён на рис. 7; сто́ит отметить, что *в нашем примере* зеркальный автомат фактически совпадает с исходным (может быть получен из него переименованием вершин) – но, вообще говоря, это, конечно, не обязательно.

Детерминированный (канонический) автомат для зеркального приведён на рис. 8; он совпадает с автоматом, приведённым на рис. 6 – но в общем случае такой факт тоже, вообще говоря, неверен. А соответствие вершин в случае зеркальных автоматов и языков таково: вершине 1 соответствуют вершины X и Y, вершине 2 – только Z, а вершине 3 – только Y.

При этом мы получили ещё более важную информацию: вершина

---

5 Например, вершина 1 соответствует вершине B в связи со следующим фактом. Существует слово, прочтя которое первый автомат может находиться в 1, а второй – в B. Таким словом может быть, например, слово из одной буквы a – и отметим, что это не единственный возможный вариант. Более подробные пояснения, а также все остальные случаи соответствия вершин не рассматриваем.

B (на рис. 6) «через» вершину 2 (на рис. 5 и 7) соответствует вершине Z (на рис. 8); аналогично вершина C – вершине Y, а каждая из вершин A и B («через» вершину 1) – каждой из вершин X и Y. Таким образом, получается *таблица соответствия вершин* автоматов, приведённых на рис. 6 и 8 (канонических для заданного и зеркального языков) – в нашем примере такая таблица приведена на рис. 9 (слева – более наглядный, справа – более формальный её варианты):

	X	Y	Z
A	#	#	
B	#	#	#
C		#	

	X	Y	Z
A	1	1	0
B	1	1	1
C	0	1	0

Рис. 9

При этом важно отметить, что каждая такая таблица соответствует даже не автомату, а задаваемому им языку; с подобными таблицами мы и будем работать в дальнейшем. Более того, можно доказать, что любая матрица (прямоугольная таблица), заполненная 0 и 1 и имеющая специальные просто формулируемые «хорошие» свойства[6], может являться таблицей соответствия вершин *некоторого* конечного автомата; см. [22, 23] и др.

Мы продолжим рассматривать наш подход к реализации метода ветвей и границ в части II настоящей статьи.

СПИСОК ЛИТЕРАТУРЫ

1. Melnikov, B. (2006), “Multiheuristic approach to discrete optimization problems”, *Cybernetics and Systems Analysis*, Vol. 42, No. 3, pp. 335–341.
2. Melnikov, B., Radionov, A., Moseev, A., Melnikova, E. (2006), “Some specific heuristics for situation clustering problems”, *Proceedings 1st International Conference on Software and Data Technologies, ICSOFT 2006*, Vol. 42, No. 3, pp. 272–279.
3. Мельников, Б. Ф., Мельникова, Е. А. (2007), «Кластеризация ситуаций и принятие решений в задачах дискретной оптимизации», *Известия высших учебных заведений. Поволжский регион*, № 2, сс. 25–29.
4. Мельников, Б. Ф., Панин, А. Г. (2012), «Параллельная реализация мультиэвристического подхода в задаче сравнения генетических последовательностей», *Вектор науки Тольяттинского государ-*

6 А именно – что в ней нет одинаковых срок, никакая строка не состоит только из 0, и оба этих условия выполняются и для столбцов.

ственного университета, № 4 (22), сс. 83–86.
5. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. (2013), *Алгоритмы: построение и анализ*, Вильямс, М., 1328 с.
6. Липский, В. (1988), *Комбинаторика для программистов*, Мир, М., 200 с.
7. Melnikov, B. (2000), "Once more about the state-minimization of the nondeterministic finite automata", *Journal of Applied Mathematics and Computing*, Vol. 7, No. 3, pp. 655–662.
8. Мельников, Б. Ф., Мельникова, Е. А. (2013), «Некоторые эвристические алгоритмы в задаче вершинной минимизации недетерминированных конечных автоматов», *Стохастическая оптимизация в информатике*, Т. 9, № 2, сс. 73–87.
9. "The 2008 ACM Programming Contest World Finals (2007)", режим доступа: https://icpc.baylor.edu/regionals/finder/world-finals-2007
10. Melnikov, B., Melnikova, E. (2007), "Some competition programming problems as the beginning of artificial intelligence", *Informatics in Education*, Vol. 6, No. 2, pp. 385–396.
11. Рассел, С., Норвиг, П., (2006), *Искусственный интеллект: современный подход*, Вильямс, М., 1408 с.
12. Люгер, Дж. (2003), *Искусственный интеллект: стратегии и методы решения сложных проблем*, Вильямс, М., 864 с.
13. Громкович, Ю. (2010), *Теоретическая информатика. Введение в теорию автоматов, теорию вычислимости, теорию сложности, теорию алгоритмов, рандомизацию, теорию связи и криптографию*, БХВ, СПб, 338 с.
14. Сазонов, А. М., Соколов, А. В., Старков, С. А. (2014), «Реализация параллельных алгоритмов решения некоторых оптимизационных задач на графах», *Эвристические алгоритмы и распределённые вычисления*, Т. 1, № 6, сс. 76–92.
15. "Microsoft Foundation Class Library Version 7.0", режим доступа: https://msdn.microsoft.com/en-us/library/37f1f848(v=VS.71).aspx
16. Ахо, А., Ульман, Дж. (1977), *Теория синтаксического анализа, перевода и компиляции. Т.1: Синтаксический анализ*, Мир, М., 612 с.
17. Polák, L. (2005), "Minimalizations of NFA using the universal automaton", *Int. J. Found. Comput. Sci.*, Vol. 16, Iss. 5, pp. 999–1010.
18. Geldenhuys, J., van der Merwe. B., van Zijl, L. (2010), "Reducing nondeterministic finite automata with SAT solvers", *Finite-State Methods and Natural Language Processing. Lecture Notes in Computer Science*, Springer. Vol. 6062, pp. 81–92.
19. Han, Y.-S. (2013), "State elimination heuristics for short regular ex-

pressions", *Fundamenta Informaticae*, Vol. 128, pp. 445–462.

20. Li, L., Qiu, D. (2015), "On the state minimization of fuzzy automata", *IEEE Transactions on Fuzzy Systems*, Vol. 23, No. 2, pp. 434–443.
21. Брауэр, В., (1987), *Введение в теорию конечных автоматов*, Радио и связь, М., 390 с.
22. Зубова, М. А., Мельников, Б. Ф. (2012), «Об одном алгоритме построения универсального автомата Конвея», *Вестник Воронежского государственного университета. Серия: Физика. Математика*, № 1, сс. 135–137.
23. Долгов, В. Н., Мельников, Б. Ф. (2013), «Построение универсального конечного автомата. I. От теории к практическим алгоритмам», *Вестник Воронежского государственного университета. Серия: Физика. Математика*, № 2, сс. 173–181.

An approach to the implementation of branch and bound method for discrete optimization problems. Part I

B. F. Melnikov, email: bf-melnikov@yandex.ru
E. A. Melnikova, email: ya.e.melnikova@yandex.ru
Samara State University

Abstract. The main purpose of the paper is the consideration of the method for easily implementation of the solutions for a class of search problems. It can be called the approach to the implementation of complex algorithms that are based on branch and bound method, which includes many more of the same type of heuristics. This uniformity is that the heuristics themselves virtually did not change, are carried out with the transition from one task to another. Therefore, we call this method by multiheuristic approach. Often, this method gives a set of heuristic algorithms for practical solutions of some NP-hard problem.

According to the authors, in many scientific publications devoted to description of the algorithms, as well as various books, the very implementation of the programs is in fact a “taboo”. Moreover, there occurs often a bad situation: the texts of programs given in a very good book seem a little “unfinished”, because the authors do not pay attention to the “pure programming” questions.

And the second purpose of this paper is considering a specific example of one of these tasks, and it includes a review of the basics of the mathematical theory required for its solution. In this paper, we give an elementary exposition of the theory of minimization of nondeterministic finite automata; rather, we consider its part, which is used for programmers, which do not need to look this area in detail.

Key words and phrases: discrete optimization problem; heuristic algorithm; programming realization; branch and bound method.

Computing Classification System 1998: G.1.2, G.2.1

Mathematics Subject Classification 2010: 68W30, 68W99

REFERENCES

1. Melnikov, B. (2006), “Multiheuristic approach to discrete optimization problems”, *Cybernetics and Systems Analysis*, Vol. 42, No. 3, pp. 335–341.
2. Melnikov, B., Radionov, A., Moseev, A., Melnikova, E. (2006), “Some specific heuristics for situation clustering problems”, *Proceedings 1st International Conference on Software and Data Technologies, ICSOFT 2006*, Vol. 42, No. 3, pp. 272–279.
3. Melnikov, B. F., Melnikova, E. A. (2007), “Clustering of cases and decision-making in discrete optimization problems”, *Izvestiya of higher education institutions. Volga region* [“Klasterizaciya situaciy i prinyatie resheniy v zadachah diskretnoy optimizacii”, *Izvestiya vysshyh uchebnyh zavedeniy. Povolzhskiy region*], No. 2, pp. 25–29.
4. Melnikov, B. F., Panin, A. G. (2012), “Parallel implementation of multiheuristic approach in the task of comparing genetic sequences”, *Vector of Science of Togliatti State University* [“Parallel’naya realizaciya mul’tievristicheskogo podhoda v zadache sravneniya geneticheskih posledovatel’nostey”, *Vektor nauki Tol’yattinskogo gosudarstvennogo universiteta*], No. 4 (22), pp. 83–86.
5. Cormen, T. H., Leiserson, Ch. E., Rivest, R. L., Stein, C. (2009), *Introduction to*

Algorithms, MIT Press, Boston., 1292 p.
6. Lipski, W. (2004), *Kombinatoryka dla programistów*, Wydawnictwa Naukowo-Techniczne, Warszawa, 274 s.
7. Melnikov, B. (2000), "Once more about the state-minimization of the nondeterministic finite automata", *Journal of Applied Mathematics and Computing*, Vol. 7, No. 3, pp. 655–662.
8. Melnikov, B. F., Melnikova, E. A. (2013), "Some heuristic algorithms in the problem of state-minimization of nondeterministic finite automata", *Stochastic optimization in informatics* ["Nekotorye evristicheskie algoritmy v zadache vershinnoy minimizacii nedeterminiovannyh konechnyh avtomatov", *Stohasticheskaya optimizaciya v informatike*], Vol. 9, No. 2, pp. 73–87.
9. "The 2008 ACM Programming Contest World Finals (2007)", access mode: https://icpc.baylor.edu/regionals/finder/world-finals-2007
10. Melnikov, B., Melnikova, E. (2007), "Some competition programming problems as the beginning of artificial intelligence", *Informatics in Education*, Vol. 6, No. 2, pp. 385–396.
11. Russell, S., Norvig, P., (2002), *Artificial Intelligence: A Modern Approach*, Prentice Hall, NJ, 1132 c.
12. Luger, G. F. (2003), *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, Addison-Wesley, Boston, 890 p.
13. Hromkovič, J. (2011), *Theoretische Informatik: Formale Sprachen, Berechenbarkeit, Komplexitätstheorie, Algorithmik, Kommunikation und Kryptographie*, Springer Verlag, Berlin, 407 S.
14. Sazonov, A. M., Sokolov, A. V., Starkov, S. A. (2014), "Implementation of parallel algorithms for the solution of optimization problems on the graphs", *Heuristic Algorithms and Distributed Computing* ["Realizaciya parallel'nyh algoritmov resheniya optimizacionnyh zadach na grafah", *Evisticheskie algoritmy i raspredelyonnye vychisleniya*], Vol. 1, No. 6, cc. 76–92.
15. "Microsoft Foundation Class Library Version 7.0", access mode: https://msdn.microsoft.com/en-us/library/37f1f848(v=VS.71).aspx
16. Aho, A. V., Ullman, J. D. (1972), *The Theory of Parsing, Translation, and Compiling, Vol. 1: Parsing*, Prentice-Hall, NJ, 848 p.
17. Polák, L. (2005), "Minimalizations of NFA using the universal automaton", *Int. J. Found. Comput. Sci.*, Vol. 16, Iss. 5, pp. 999–1010.
18. Geldenhuys, J., van der Merwe. B., van Zijl, L. (2010), "Reducing nondeterministic finite automata with SAT solvers", *Finite-State Methods and Natural Language Processing. Lecture Notes in Computer Science*, Springer. Vol. 6062, pp. 81–92.
19. Han, Y.-S. (2013), "State elimination heuristics for short regular expressions", *Fundamenta Informaticae*, Vol. 128, pp. 445–462.
20. Li, L., Qiu, D. (2015), "On the state minimization of fuzzy automata", *IEEE Transactions on Fuzzy Systems*, Vol. 23, No. 2, pp. 434–443.
21. Brauer, W., (1984), *Automatentheorie. Eine Einführung in die Theorie endlicher Automaten*, Springer, Berlin, 491 S.
22. Zubova, M. A., Melnikov, B. F. (2012), "On an algorithm of constructing Conway's universal automaton", *Vestnik of Voronezh State University Series: Physics. Mathematics* ["Ob odnom algoritme postroeniya universal'nogo avtomata Konveya", *Vestnik Voronezhskogo gosudarstvennogo universiteta. Seriya: Fizika. Matematika*], No. 1, pp. 135–137.

23. Dolgov, V. N., Melnikov, B. F. (2013), “Constructing universal finite automaton. I. From the theory to the practical algorithms”, *Vestnik of Voronezh State University. Series: Physics. Mathematics* [“Postroenie universal’nogo konechnogo avtomata. I. Ot teorii k prakticheskim algoritmam”, *Vestnik Voronezhskogo gosudarstvennogo universiteta. Seriya: Fizika. Matematika*], No. 2, pp. 173–181.

Эвристические алгоритмы и распределённые вычисления, **2,** 4 (2015) 73–92

УДК 004.852

Квадратичная проективная регрессия как метод обучения в разрежённых пространствах высокой размерности *

А. А. Сенов, email: alexander.senov@gmail.com
Санкт-Петербургский государственный университет

Аннотация. Проблема обучения с учителем в разрежённых пространствах высокой размерности пространства входов становится всё более актуальной в последние годы. К сожалению, большинство классических методов, используемых в задачах с малой размерностью пространства входов, неприменимы с случае высокой размерности из-за низкого качества и высокой вычислительной сложности. Так, линейные методы хоть и обладают низкой вычислительной сложностью, но демонстрируют низкое качество предсказания, так как неспособны учесть сложные зависимости между признаками. Нелинейные методы, напротив, учитывают зависимости между признаками более чем первого порядка, но их вычислительная сложность при этом обычно растёт экспоненциально.

Для борьбы с так называемым «проклятием размерности» традиционно применяются методы, уменьшающие размерность пространства входов с помощью тех или иных преобразований. Существенным недостатком подобных методов является то, что они не учитывают значения выходов модели, рискуя таким образом при снижении размерности отбросить важную для предсказания значения выходов модели информацию.

Одним из важных результатов в задаче обучения в разрежённых пространствах стал предложенный в 1981 году метод Projective Pursuit Regression, состоящий из последовательного применения линейной проекции и непараметрического оценивания. Ещё одним важным шагом стал предложенный в 2010 метод Factorization Machines, моделирующий зависимости от комбинаций признаков не напрямую, а через скалярное произведение соответствующих им факторных векторов. Таким образом, Factorization

Machines одновременно даёт возможность оценивать зависимости более чем первого порядка и при этом избегает экспоненциального роста числа параметров и вычислительной сложности.

В настоящей работе для решения задачи обучения с учителем в разрежённых пространствах высокой размерности предлагается метод квадратичной проективной регрессии, идеологически схожий с Projective Pursuit Regression. Важным отличием является то, что после операции линейной проекции вместо непараметрической функции используется квадратичная, тем самым упрощается вычисление и моделируются взаимосвязи между признаками второго порядка. В работе оценивается вычислительная сложность метода квадратичной проективной регрессии. Наконец, превосходство метода квадратичной проективной регрессии над Factorization Machines определяется посредством эксперимента на реальных данных.

Ключевые слова и фразы: стохастическая оптимизация; высокая размерность; квадратичная регрессия; проективные методы.

1 Введение

В настоящее время различные алгоритмы машинного обучения получают всё более широкое применение в различных областях: распознавании образов, текстовом поиске, управлении беспилотными автомобилями и летательными средствами, противомошеннических системах, кредитном скоринге, рекомендательных системах [1, 2]. В особенности это относится к такому подразделу машинного обучения, как обучение с учителем. Задача обучения с учителем заключается в восстановлении функциональной зависимости между объектами входов и объектами выходов на основании так называемого тренировочной (обучающей) выборки — множества пар (объект входа, объект выхода). При этом на практике в качестве объектов входа и выхода рассматриваются не элементы произвольных множеств, а векторы и скаляры соответственно [3]. При этом размерности вектора входов в литературе, связанной с машинным обучением и анализом данных, принято называть *признаками*.

В последние годы всё большее распространение получают задачи, в которых размерность пространства входов крайне высока; она достигает 10^6 и более [1, 4]. Стоит отметить, что рост размерности пространства входов часто не связан с качественным ростом содержащейся в них информации: обычно лишь крайне небольшое число элементов векторов входов отличны от нуля. Отчасти это связа-

но с применением такого метода векторного кодирования объектов, как “one hot encoding” [5], который представляет каждую качественную характеристику объекта в виде многомерного вектора, чья размерность равна количеству допустимых значений признака, и лишь один элемент которого отличен от нуля. Если рассмотреть такую категориальную характеристику, как идентификатор пользователя, то причина появления векторов размерности 10^6 станет вполне очевидной.

Высокая размерность делает трудноприменимыми многие алгоритмы обучения, в особенности те, которые учитывают зависимости признаков более чем первого порядка. В частности, это связано с экспоненциальным ростом вычислительной сложности некоторых алгоритмов вместе с ростом размерности пространства входов, или же с деградацией качества предсказания алгоритмов. Комплекс проблем, возникающий в многомерных пространствах, принято называть «проклятием размерности» [1, 4]. Таким образом, крайне актуально видится развитие методов обучения, которые, с одной стороны, способны учитывать зависимости выходной переменной от признаков более чем первого порядка, а с другой стороны, могут быть эффективно применены в случае высокой размерности пространства входов.

В 2010 году был предложен метод обучения с учителем “Factorization Machines” [6], учитывающий зависимости второго порядка за счёт эффективного использования разрежённости пространства входов. Данный метод продемонстрировал простоту реализации и интерпретации, а также превосходство над некоторыми другими алгоритмами в смысле точности предсказания, такими как метод опорных векторов (SVM) и метод ближайших соседей (KNN) [6, 7].

В настоящей работе предложен новый алгоритм квадратичной проективной регрессии, предназначенный для эффективного обучения в разрежённых пространствах высокой размерности и способный при этом учитывать зависимости второго порядка. Метод основан на последовательном применении операций линейной проекции и квадратичной регрессии, он рассматривается как альтернатива методу Factorization Machines как одному из наиболее эффективных методов обучения в задачах высокой размерности как в смысле точности предсказания, так и в смысле вычислительной сложности.

Статья организована следующим образом. В разделе 2 рассматри-

вается постановка задачи обучения с учителем и классические методы её решения. В разделе 3 описываются основные причины и примеры высокой размерности и разрежённости пространства входов. В разделе 4 описываются проблемы, возникающие у большинства методов обучения с учителем в разрежённых пространствах высокой размерности, и описывается алгоритм Factorization Machines. В разделе 5 приводится описание модели квадратичной проективной регрессии, алгоритма оценки параметров модели методом стохастического градиентного спуска, а также анализ его вычислительной сложности. В разделе 6 приводится экспериментальное сравнение алгоритмов проективной регрессии и Factorization Machines на основе практической задачи обучения. Наконец, в разделе 7 делаются выводы и предлагаются дальнейшие направления исследований.

2 Обучение с учителем

2.1 Постановка задачи

Классически задача обучения с учителем заключается в восстановлении функциональной зависимости $f : \mathbb{R}^p \to \mathcal{T}$ на основе обучающего множества (тренировочного, эталонного, множества прецедентов) независимых одинаково распределённых наблюдений (пар вход-выход): $Z = \{x^{(i)}, y^{(i)}\}_1^N$ [3]. В качестве $\mathcal{T}$ обычно рассматривается либо множество $\mathbb{R}$ — тогда подобную задачу называют задачей (восстановления) *регрессии*, либо множество $\{0, 1\}$ — тогда задачу называют задачей *двуклассовой* (0 *и* 1) *классификации*. Требуется найти такую функциональную зависимость $f(x)$, чтобы ошибка на Z, называемая *эмпирическим риском*, была минимальна [3]:

$$\mathcal{L}(f|Z) = \sum_{(x,y)\in Z} \ell(y, f(x)) \to \min_f, \qquad (1)$$

где ℓ — функция ошибки на одном наблюдении, выбираемая в зависимости от типа выходного пространства $\mathcal{T}$. Так, для задачи регрессии наиболее распространённой является квадратичная функция ошибки:

$$\ell(y, f(x)) = (f(x) - y)^2,$$

а для задачи классификации — логистическая функция потерь:

$$\ell(y, f(x)) = \log\left(1 + e^{-yf(x)}\right).$$

2.2 Обзор методов обучения с учителем

Существует множество методов обучения с учителем, как для регрессии (линейная регрессия, искусственная нейронная сеть, K ближайших соседей, полиномиальная регрессия, ядерные оценки), так и для классификации (логистическая регрессия, метод опорных векторов, наивный байесовский классификатор, деревья решений).

Стоит отметить, что большинство алгоритмов обучения с учителем состоят из:

- параметрической модели функциональной зависимости y от x: $\widehat{y} = f(x|\theta)$.
- метода поиска оптимального значения параметра, доставляющего минимум функции потерь (1): $\widehat{\theta} = \operatorname*{argmin}_{\theta} \mathcal{L}(f(\cdot, \theta)|Z)$.

С точки зрения вида функции $f(x|\theta)$ методы обучения с учителем делятся на два основных типа [4].

- Линейные методы, методы первого порядка – в которых различные признаки (измерения вектора x) входят в модель (функцию $f(x|\theta)$) в виде линейной комбинации, т.е. функция f представима в виде
$$f(x|\theta) = f'(\sum_{i=1}^{p} x_i\beta_i|\alpha), \quad \text{где } \theta = (\beta_1, \dots, \beta_p, \alpha).$$
- Нелинейные методы, методы более чем первого порядка, не являющиеся линейными – способные учитывать не только индивидуальные значения признаков (измерений вектора x), но и их комбинации.

Одним из наиболее распространённых линейных методов является линейная регрессия, модель которой записывается следующим образом:
$$f(x|\theta, \theta_0) = \theta_0 + x^T\theta_1 .$$

Линейные методы обладают несколькими преимуществами: они легко поддаются интерпретации, имеют сравнительно низкую вычислительную сложность, а в некоторых случаях допускают явное выражение оптимального значения параметра $\widehat{\theta} = \operatorname{argmin} \mathcal{L}$ как функции от Z (как, например, метод наименьших квадратов [8]).

Однако линейные методы обладают существенным недостатком: они неспособны учесть нелинейные зависимости между входом x и выходом y, и, тем самым, в большинстве случаев не могут достичь оптимальной точности.

Таким образом, линейные методы неприменимы в задачах, где имеют место высокие требование к точности предсказания. В этой работе мы сконцентрируемся на рассмотрении именного такого класса задач: задач, в которых необходимо учитывать нелинейные зависимости от признаков, – зависимости второго порядка.

3 Разрежённые пространства высокой размерности

В связи с постоянным ростом объема данных всё более остро встаёт проблема обучения в условиях высокой размерности пространств векторов входов. Одной из причин высокой размерности является использование такой техники кодирования объектов входов, как “One Hot Encoding” (OHE). OHE используется в случаях, когда объект входа обладает множеством качественных (категориальных) характеристик, и заключается в следующем: все качественные характеристики объектов входов вместе с их возможными значениями нумеруются и каждому объекту сопоставляется вектор x, где $x_i = 1$, если соответствующая характеристика объекта равна соответствующему значению, и $x_i = 0$, если нет.

Приведём несколько распространенных примеров использования техники One Hot Encoding.

1. Обработка текстов.
   Для использования в документов в тех или иных алгоритмах обучения их сперва необходимо представить в векторном в виде. One Hot Encoding здесь применяется следующим образом: слова рассматриваются как категориальная характеристика (а каждое конкретно слово – её значение), и каждый документ кодируется в виде вектора x, чьи ненулевые значения соответствуют содержащимся в нём словам.

2. Рекомендательные системы.
   Одна из задач рекомендации – определить вероятность того, что конкретный пользователь купит конкретный товар. В

данном случае в качестве объекта входа рассматривается пара (пользователь, товар), пользователи нумеруются числами $1 \le i \le N$, а товары – $1 \le j \le M$ (N – количество пользователей, M – количество товаров). Тогда объект (пользователь, товар) кодируется в виде вектора x, где

$$x_l = \begin{cases} 1, & l \in \{i, N + j\}, \\ 0, & \text{в остальных случаях}, \end{cases}$$

где i – номер пользователя, а j – номер товара.

3. Категориальные признаки.
   В случае, когда объект входа обладает категориальным признаком, принимающим C различных значений (например, цвет можно рассмотреть как категориальный признак, принимающий одно из семи значений: красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый), его можно закодировать в виде вектора размерностью C:

$$x_l = \begin{cases} 1, & \text{признак принял значение номер } l \le C - 1, \\ 0, & \text{в остальных случаях}. \end{cases}$$

One Hot Encoding естественным образом применяется для преобразования объектов векторном виде во множестве задач, связанных с машинным обучением. Из использования этой техники кодирования следует два важных факта.

1. Размерность векторного представления объекта крайне высока. Так в первом примере она равна размеру словаря (p может достигать 10^6), во втором – количеству пользователей и товаров (p может достигать 10^7).

2. Полученные векторы имеют разрежённую структуру – их значение отлично от нуля лишь в небольшом количестве измерений. Так, в случае текстов число ненулевых значений равно числу уникальных слов в тексте, а в случае рекомендаций оно равно 2.

Из этих особенностей следует тот факт, что большинство методов обучения с учителем неприменимо из-за вышеупомянутого проклятия размерности [4, 9] – их вычислительная сложность вместе с

ростом размерности p растёт экспоненциально. Подобная проблема свойственна, например, для метода ближайших соседей, деревьев решений, ядерных оценок, полиномиальной регрессии. С другой стороны, данные имеют разрежённую структуру – реальная размерность каждого вектора входа гораздо меньше фактической, что потенциально может быть использовано при обучении.

4 Обучения с учителем в разрежённых пространствах высокой размерности

В этом разделе мы рассмотрим особенности обучения в пространствах высокой размерности. Так, мы покажем, какие именно сложности связаны с применением классических нелинейных методов (квадратичной линейной регрессии) и опишем метод Factorization Machines, который на данный момент является одним из передовых в задаче обучения с учителем в разрежённых пространствах высокой размерности.

4.1 Квадратичная линейная регрессия

Модель квадратичной регрессии записывается следующим образом:

$$f_{SLR}\left(x|\{\alpha, \beta, \omega_{i,j}\}_{1\leq i\leq j\leq p}\right) = \alpha + \sum_{i=1}^{p} x_i\beta_i + \sum_{1\leq i\leq j\leq p} x_i x_j \omega_{i,j}\,.$$

В задачах с высокой размерностью квадратичная линейная регрессия требует оценки порядка $O(p^2)$ параметров, что может быть крайне затруднительно. Более того, в силу разрежённости векторов входов некоторые параметры модели просто не могут быть достоверно оценены, потому что в обучающей выборке соответствующие комбинации ненулевых признаков не встречаются или встречаются крайне редко. Так, например, параметр $\omega_{i,j}$ не может быть достоверно определён, если в обучающей выборке нет векторов входов x, у которых $x_i \neq 0$ и $x_j \neq 0$.

4.2 Метод Factorization Machines

В работе [6] была предложена нелинейная модель обучения с учителем Factorization Machines (далее FM), сопоставляющая каждому

признаку (измерению) вектора входов вектор факторов. Таким образом, в методе FM через скалярные произведения этих факторных векторов моделируются нелинейные взаимодействия между признаками.

Модель FM записывается следующим образом:

$$f_{FM}(x|\omega_0, \omega_1, \nu_1, \dots, \nu_p) = \alpha + x^T\beta + \sum_{1<i<j<p} <\nu_i, \nu_j> x_i x_j, \quad (2)$$

где $\alpha \in \mathbb{R}$, $\beta \in \mathbb{R}^p$, $\nu_i \in \mathbb{R}^K$ $\forall i = 1..p$, $K << p$.

Как видно, модель крайне проста и состоит из двух типов слагаемых:

- $\alpha + x^T\beta$ – линейное слагаемое, соответствующее линейной регрессии, описывающее линейную зависимость между y и признаками x_i;
- $< \nu_i, \nu_j > x_i x_j$ – квадратичное слагаемое, описывающие влияние между y и $x_i x_j$ (взаимодействие признаков x_i и x_j).

Стоит отметить, что особенностью и основным преимуществом FM является то, что взаимодействие признаков x_i и x_j выражается не явно через параметр $\omega_{i,j}$ (как это делается, например, в квадратичной регрессии), а неявно через скалярное произведение соответствующих векторов факторов ν_i и ν_j. Подобный приём аналогичен методам матричной факторизации, где оригинальная матрица приближается произведением двух (и более) *факторных матриц*, таким образом каждый её элемент представляется в виде произведения двух (и более) столбцов факторных матриц.

Выделим основные преимущества данного подхода (согласно [6]).

- Количество параметров линейно зависит от размерности пространства входов.
- Решается проблема малого числа измерений с ненулевым произведением $x_i x_j$. Поскольку значение фактора ν_i оценивается на всех наблюдениях, где $x_i \neq 0$, это позволяет получить значение оценку коэффициента взаимодействия x_i и x_j ($< \nu_i, \nu_j >$) даже в случае отсутствия наблюдений, для которых $x_i \neq 0$ и $x_j \neq 0$ одновременно.

- Низкая вычислительная сложность. В силу простоты взаимодействия факторов ν_i, ν_j (через скалярное произведение), значение функции $f_{FM}(x|\cdot)$ может быть вычислено за $\mathcal{O}\left(\mu_x^2 K\right)$, а значение её градиента по ν_i – за $\mathcal{O}\left(\mu_x^2 K\right)$, где μ_x – количество измерений x, отличных от нуля.

В работе [7] были предложены методы оценки параметров f_{FM} на основе стохастического градиентного спуска, чередующего метода наименьших квадратов (alternating least squares) и метода Монте-Карло по цепями Маркова. Так, алгоритм оценки на основе стохастического градиентного спуска имеет вычислительную сложность порядка $\mathcal{O}\left(TK\mu_{avg}\right)$, где T – количество итераций, K – количество факторов, а μ_{avg} – среднее количество ненулевых элементов в векторах входов x.

В следующем разделе мы предложим метод, обладающий преимуществами как схожими с Factorization Machines, так и некоторыми дополнительными.

5 Метод квадратичной проективной регрессии

В этом разделе мы опишем метод квадратичной проективной регрессии, рассмотрим его модель, способ оценки ее параметров посредством стохастического градиентного спуска (далее SGD – stochastic gradient descent), а также обсудим его теоретические и вычислительные свойства.

5.1 Модель квадратичной проективной регрессии

Начнём с модели квадратичной проективной регрессии. Она записывается следующим образом.

$$f(x|\{\alpha, \beta_0, \dots, \beta_K\}) = \alpha + x^T\beta_0 + \sum_{k=1}^{K} \left(x^T\beta_k\right)^2, \tag{3}$$

где $\alpha \in \mathbb{R}$, $\beta_0 \in \mathbb{R}^p$, $\beta_k \in \mathbb{R}^p$ $\forall k = 1..K$, $K << p$.

Модель состоит из следующих слагаемых:

- $\alpha + x^T \beta_0$ – линейное слагаемое;

- $(x^T \beta_k)^2$ – квадратичные проективные слагаемые, которые в свою очередь является композицией двух функций:

  – $x^T \beta_k$ – проекции, за счёт которой решается проблемы высокой размерности,

  – $(\cdot)^2$ – квадратичной функции, за счёт которой достигается нелинейность и становится возможным учитывать взаимосвязи признаков.

Оригинальным в этой модели являются именно квадратичные проективные слагаемые. Как мы упоминали ранее, квадратичная регрессия неприменима в задачах с высокой размерностью, так как требует оценки порядка $O(p^2)$ параметров. В квадратичной проективной регрессии эта проблема решается тем, что размерность фактически снижается с p до K за счёт проекции, и единственные параметры, которые требуется оценить (за исключением участвующих в линейной комбинации) – это параметры проекции $\beta_1, \ldots, \beta_K$.

Стоит отметить, что модель квадратичной проективной регрессии схожа с моделью Projection Pursuit Regression [10] – с тем лишь отличием, что вместо непараметрических оценок плотности используется квадратичная функция. Кроме того, важным отличием является область применения. Projection Pursuit Regression используется преимущественно для задач, где векторы входов имеют относительно небольшую размерность и мало нулевых значений, в то время как квадратичная проективная регрессия нацелена на задачи, где векторы входов разрежены и имеют высокую размерность.

Кроме того, такой подход схож с подходом, используемым в методах снижения размерности. Такие методы, как PCA, ICA, MDS используют линейные и нелинейные проекции векторов входов из пространства высокой в пространство низкой размерности. Важным отличием является то, что проекции, которые находит квадратичная проективная регрессия, как будет показано далее, выбираются из соображений минимизации функции ошибки (1), в то время как методы снижения размерности не учитывают значения выходной переменной при построении проекций.

5.2 Оценка параметров квадратичной проективной регрессии

В силу того, что функция ошибки $\mathcal{L}$ выпукла по параметрам $\{\alpha, \beta_0, \ldots, \beta_K\}$, для определения параметров может быть использован любой метод выпуклой оптимизации. Задачи, связанные с высокой размерностью, часто сопряжены и с бо́льшим числом измерений N (как, например, в задаче рекомендаций, или обработки текстов), поэтому целесообразным является использование такого метода, как стохастический градиентный спуск (далее, SGD – stochastic gradient descent), относящегося к более широкому классу методов стохастической аппроксимации [11]. Особенностью SGD является то, что он экономит вычислительные ресурсы в случае большого числа измерений и де-факто является стандартом в задачах машинного обучения с большим количеством измерений (т.н. large scale learning) [12].

Основная идея SGD заключается в последовательном рассмотрении наблюдений $(x, y) \in Z$ и движении против градиента функции ошибки на этом наблюдении:

$$\theta^{(t)} = \theta^{(t-1)} - \eta \nabla_\theta \left(x | \theta^{(t-1)} \right),$$

где θ – параметры модели ($\{\alpha, \beta_0, \ldots, \beta_K\}$), η – шаг алгоритма, а t – номер итерации.

Приведённый ниже алгоритм 1 представляет более подробное описание процедуры оценки параметров квадратичной проективной регрессии посредством стохастического градиентного спуска, выполняемого поочерёдно по каждому из параметров $\{\alpha, \beta_0, \ldots, \beta_K\}$. Стоит отметить, что он явно использует разрежённость векторов входов для уменьшения числа операций посредством $\mathcal{I}$ – списка ненулевых признаков (измерений) вектора x.

5.3 Свойства квадратичной проективной регрессии

Приведём несколько утверждений относительно вычислительной сложности квадратичное проективной регрессии.

Замечание 1. Пусть число ненулевых элементов x равно в среднем μ_{mean}, доступ по индексу к элементам векторов $\{\beta_k\}_0^K$ и x выполняется за $\mathcal{O}(1)$, а список индексов ненулевых значений вектора x $\mathcal{I} = \{i \in \{1..p\} : x_i \neq 0\}$ может быть получен за $\mathcal{O}(|\mathcal{I}|)$.

Алгоритм 1 *SquaredProjectiveRegression_via_SGD* (
: $Z = \{x_i, y_i\}_1^N$ — обучающее множество,
: K — количество проекций,
: T — число итераций,
: η — шаг SGD,
: λ — коэффициент l_2- регуляризации
)

$\alpha_0^{(0)} \leftarrow 0$
$\beta_0^{(0)} \leftarrow 0_p$
for all $k \leftarrow 1$ to K **do**
  $\beta_k^{(0)} \leftarrow \mathcal{N}(0,1)$
end for
for all $t \leftarrow 1$ to T **do**
  Draw $(x, y) \in Z$ at random
  $\mathcal{I} \leftarrow \{i \in \{1..p\} : x_i \neq 0\}$
  $\ell^{(t)} \leftarrow \left(\alpha_0^{(t-1)} + x^\intercal \beta_0^{(t-1)} + \sum_{k=1}^K \left(x^\intercal \beta_k^{(t-1)}\right)^2\right) - y$
  $\alpha_0^{(t)} \leftarrow \alpha_0^{(t-1)} - \eta \left(2\ell^{(t)} + 2\lambda\alpha_0^{(t-1)}\right)$
  for all $i \leftarrow \mathcal{I}$ **do**
    $\beta_{0,i}^{(t)} \leftarrow \beta_{0,i}^{(t-1)} - \eta \left(2\ell^{(t)} x_i + 2\lambda\beta_{0,i}^{(t-1)}\right)$
  end for
  for all $k \leftarrow 1$ to K **do**
    for all $i \leftarrow \mathcal{I}$ **do**
      $\beta_{k,i}^{(t)} \leftarrow \beta_{k,i}^{(t-1)} - \eta \left(2\ell^{(t)} \left(x^\intercal \beta_k^{(t-1)}\right) x_i + 2\lambda\beta_{k,i}^{(t-1)}\right)$
    end for
  end for
end for
return $\{\alpha, \beta_0, \dots, \beta_K\}$

Тогда вычислительная сложность модели (3) есть в среднем $\mathcal{O}(K\mu_{mean})$.

Замечание 2. Пусть число ненулевых элементов x не превышает μ_{max}, доступ по индексу к элементам векторов $\{\beta_k\}_0^K$ и x выполняется за $\mathcal{O}(1)$, а список индексов ненулевых значений вектора x $\mathcal{I} = \{i \in \{1..p\} : x_i \neq 0\}$ может быть получен за $\mathcal{O}(|\mathcal{I}|)$.

Тогда вычислительная сложность модели (3) не превышает $\mathcal{O}(K\mu_{max})$.

Замечания 1 и 2 очевидным образом следуют из того, что скалярное произведение двух векторов, один из которых имеет ненулевые значения лишь на индексах $\mathcal{I}$, может быть получено за $\mathcal{O}(|\mathcal{I}|)$.

Используя аналогичные идеи, можно доказать следующее утверждение.

Утверждение 1 *Пусть число ненулевых элементов* x *равно в среднем* μ_{avg}*, доступ по индексу к элементам векторов* $\{\beta_k\}_0^K$ *и* x *выполняется за* $\mathcal{O}(1)$*, а список индексов ненулевых значений вектора* x $\mathcal{I} = \{i \in \{1..p\} : x_i \neq 0\}$ *может быть получен за* $\mathcal{O}(|\mathcal{I}|)$*.*

Тогда вычислительная сложность алгоритма 1 есть в среднем $\mathcal{O}(TK\mu_{avg})$*.*

Доказательство. Операции вне цикла по t не оказывают существенного влияния на асимптотическую сложность. Поэтому разберём каждую операцию внутри цикла согласно номеру строки:

8 – выполняется за $\mathcal{O}(1)$ по согласно условиям;

9 – выполняется за $\mathcal{O}(K\mu_{avg})$ согласно замечанию 1;

10 – выполняется за $\mathcal{O}(1)$ как операция над скалярами;

12 – выполняется за $\mathcal{O}(1)$ как операция над скалярами;

16 – выполняется за $\mathcal{O}(1)$ как операция над скалярами (значения $x^T\beta_k^{(t-1)}$ уже вычислены в строке 9 и могут не считаться повторно).

Так как на одной итерации операция на строке 8 выполняется 1 раз, на строке 9 – 1 раз, на строке 10 – 1 раз, на строке 12 – K раз, а на строке 16 – $K\mu_{avg}$ раз, то общая вычислительная сложность одной итерации в среднем есть $\mathcal{O}(K\mu_{max})$. Отсюда вычислительная сложность алгоритма в среднем $\mathcal{O}(TK\mu_{avg})$. □

Замечание 3. Аналогичным образом может быть доказано и утверждение с ограничением на максимальное количество ненулевых элементов μ_{max}.

Таким образом, вычислительная сложность модели и алгоритма оценки параметров квадратичной проективной регрессии совпадает с вычислительной сложностью модели и алгоритма оценки параметров модели Factorization Machines.

6 Практическое сравнение с методом Factorization Machines

В этом разделе мы проведем практическое сравнение квадратичное проективной регрессии с методом Factorization Machines. Для

сравнения мы использовали данные MovieLens-100K [13], собранные исследовательской группой GroupLens. Эти данные содержат информацию о $100\,000$ рейтингах, проставленных пользователями интернет-сервиса MovieLens в период с 19 сентября 1997 г. по 22 апреля 1998 г. Рейтинги организованы как записи вида

(идентификатор пользователя, идентификатор кинофильма, рейтинг, время),

разделённые запятой; при этом значение рейтинга принимает значения от 1 до 5.

Для сравнения методов Factorization Machines и квадратичной проективной регрессии они были реализованы на языке программирования Python [14] с использованием библиотеки SciPy [15] для операций над разрежёнными данными. Для оценки параметров обеих моделей использовался стохастический градиентный спуск – см. алгоритм 1 из [7] и рассмотренный нами выше алгоритм 1 (из нашей статьи) соответственно. При этом оба метода использовались следующим образом.

1. Каждый рейтинг из данных MovieLens был преобразован в пару вход-выход x, y, где y – значение рейтинга, а x – результат One Hot Encoding на паре (идентификатор пользователя, идентификатор кинофильма); при этом детали применения OHE для задачи рекомендаций см. в разделе 3. Таким образом, размерность пространства входов составила $d = 2623$, $\mu_{avg} = 2$, $\mu_{max} = 2$.

2. Множество $Z = \{x, y\}$ случайным образом разделено на тренировочную и т.н. валидационную части: Z_{train} и Z_{valid} в пропорции $9:1$.

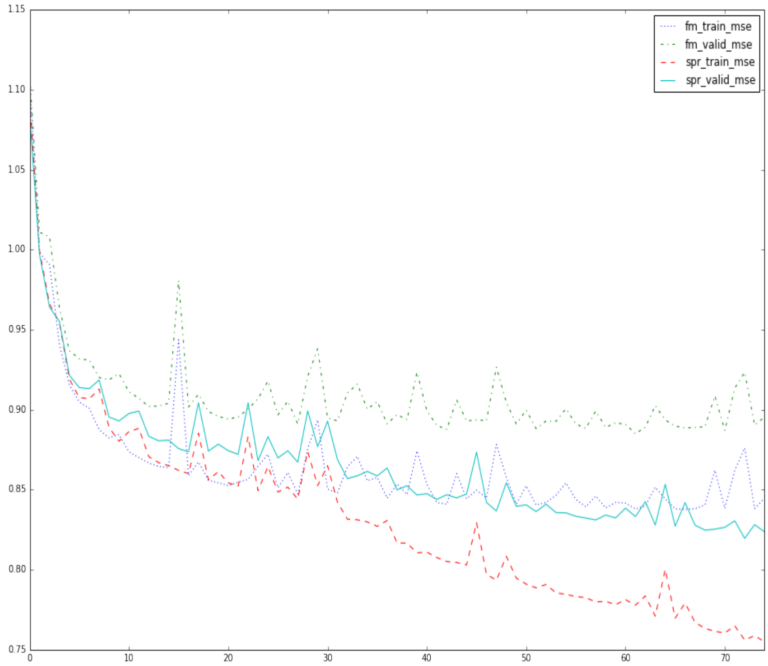
3. Параметры моделей Factorization Machines и квадратичной проективной регрессии были оценены на множестве Z_train с использованием следующих значений параметров:

   - $K = 10$,
   - $T = 700000$,
   - $\eta = 0.01$,
   - $\lambda = 0.01$.

4. Каждые $10\,000$ итераций ошибка предсказания обеих моделей оценивалась на валидационном Z_{valid} и тренировочном Z_{train} множествах. В качестве ошибки был выбрана среднеквадратичная ошибка MSE (Mean Squared Error):

$$\mathsf{MSE}\left(\left\{y^{(i)}\right\}_{i\in\mathcal{I}}, \left\{y^{(i)}_{pred}\right\}_{i\in\mathcal{I}}\right) = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\left(y^{(i)} - y^{(i)}_{pred}\right)^2,$$

где $\mathcal{I}$ – множество индексов, соответствующее Z_{valid} или Z_{train}.



На приведённом рисунке изображён график среднеквадратичной ошибки по мере тренировки обеих моделей как на тренировочной, так и для валидационной выборках согласно описанной выше схеме. При этом явно видно превосходство квадратичной проективной регрессии над Factorization Machines: начиная с итерации $50\,000$ ошибка первого в среднем меньше на 0.06, что в процентном соотношении означает улучшение на 6.8%. Кроме того, как можно заметить, последние $200\,000$ итераций ошибка на валидационной выборке у обоих методов не уменьшается при почти монотонном убывании ошибки на тренировочной выборке. Таким образом обучение практически остановилось и имеет место ситуация т.н. «переобучения» (overfitting), исходя из чего рассмотрение последующих итераций излишне.

7 Заключение

В этой работе мы рассмотрели задачу обучения с учителем в условиях разрежённого пространства входов высокой размерности, а также связанные с этим вопросы сложности, которые делают большинство нелинейных методов обучения с учителем неприменимыми. Для решения таких проблем мы предложили алгоритм квадратичной регрессии, вычислительная сложность обучения которого зависит от размерности пространства входов p линейно, а сложность предсказания значения выхода y по данному вектору входа x зависит лишь от числа ненулевых элементов в векторе.

Для оценки параметров модели квадратичной проективной регрессии нами предложен алгоритм на основе стохастического градиентного спуска и оценена его вычислительная сложность. Наконец, проведено экспериментальное сравнение метода квадратичной проективной регрессии с другим передовым методом обучения в разрежённых пространствах высокой размерности – Factorization Machines. Эксперименты показали сравнимое время выполнения обоих алгоритмов при явном преимуществе квадратичной проективной регрессии в качестве предсказания.

В дальнейшем мы планируем сделать несколько усовершенствований алгоритма. Так, планируется исследовать возможность применения других, в том числе рандомизированных алгоритмов для оценки параметров модели вместо стохастического градиентного спуска. Например алгоритм Роббинса-Монро, или рандомизированный алгоритм стохастической аппроксимации (РАСА), который особенно эффективен в случае помех в наблюдениях [8]. Кроме того, планируется исследовать возможность использования , более сложных чем квадратичная, в том числе параметрических функций, для аппроксимации значения выхода y проекциями x.

Список литературы

1. Smola, A., Vishwanathan, S. V. N. (2008), *Introduction to machine learning*, Cambridge University Press, UK, 226 p. ⇒ 74, 75
2. Witten, I. H., Frank, E. (2005), *Data Mining: Practical machine learning tools and techniques, 3rd edition*, Morgan Kaufmann, Waltham, US, 664 p. ⇒ 74

3. Вапник, В. Н., Червонекис, А. Я. (1974), *Теория распознавания образов*, Наука, Москва, 416 с. ⇒74, 76
4. Friedman, J., Hastie, T., Tibshirani, R. (2009), *The Elements of Statistical Learning, 2nd edition*, Springer, Berlin, 745 p. ⇒ 74, 75, 77, 79
5. Harris, D., Harris, S. (2012), *Digital design and computer architecture, 2nd edition*, Morgan Kaufmann, Waltham, US, 712 p. ⇒75
6. Rendle, S. (2010), "Factorization machines", *Proc. 10th IEEE Conf. On Data Mining (ICDM) (Sydney, Australia)*, pp. 995–1000. ⇒ 75, 80, 81
7. Rendle, S. (2012), "Factorization machines with libFM", *ACM Transactions on Intelligent Systems and Technology (TIST)*, Vol. 3, No. 3, pp. 57–79. ⇒75, 82, 87
8. Граничин, О. Н., Поляк, Б. Т. (2003), *Рандомизированные алгоритмы оценивания и оптимизации при почти произвольных помехах*, Наука, М., 290 с. ⇒77, 89
9. Friedman, J. H. (1997), "On bias, variance, 0/1–loss, and the curse-of-dimensionality", *Data mining and knowledge discovery*, Vol. 1, No. 1, pp. 55–77. ⇒79
10. Friedman, J. H., Stuetzle, W. (1981), "Projection pursuit regression", *Journal of the American statistical Association*, Vol. 76, No. 376, pp. 817–823. ⇒83
11. Граничин, О. Н. (2003), *Введение в методы стохастической оптимизации и оценивания. Учебное пособие*, Изд-во С.-Петерб. университета., 131 с. ⇒84
12. Bottou, L. (1981), "Large-scale machine learning with stochastic gradient descent", *Proc. COMPSTAT (Paris, France)*, pp. 177–186. ⇒84
13. GroupLens Research (2015), *European Football Championship 2012*, available at: http://grouplens.org/datasets/movielens/ ⇒87
14. Van Rossum, G., Drake, F. L. (2015), *Python Reference Manual*, available at: http://www.python.org ⇒87
15. Jones, E., Oliphant, T., Peterson, P. (2015), {*SciPy*}*: Open source scientific tools for* {*Python*}, available at: http://www.scipy.org ⇒ 87

Quadratic projective regression as a learning method in sparse high-dimensional spaces

A. A. Senov, email: alexander.senov@gmail.com
Russia, Saint-Petersburg, Saint-Petersburg State University

Abstract. Supervised learning in high-dimensional sparse spaces becoming increasingly important nowadays. Unfortunately, most classic supervised learning methods are inapplicable in high-dimensional spaces due to decreasing accuracy and increasing computational complexity. Thus, even though linear methods has low computational complexity, their show poor prediction quality because they can not take into account complex dependencies between features. On the other hand, non-linear methods, take complex dependencies into account, but their computational complexity grows exponentially together with it.
Dimensionality reduction techniques is the most common tool against the curse of dimensionality, which consist in reducing the dimensionality of input space by some kind of transformation, e.g. projection. Major drawback of such methods is that they does not take model output values into account, thus loosing some information which is important for prediction.
One the main results in supervised learning in sparse high-dimensional spaces problem is Projective Pursuit Regression method. It consists of consecutive operations of linear projection and non-parametric estimation. Another important result is a Factorization Machines method, which modeling interaction between features not explicitly, but via scalar product of corresponding factor vectors. Thus, Factorization Machines obtain both low computational complexity and ability to estimate inter-features dependencies.
In this paper we propose a method called quadratic projective regression for supervised learning in sparse high-dimensional spaces problem. This method share the same idea with Projective Pursuit Regression method, but uses quadratic function in the last step instead of non-parametric one. We show theoretical estimate of computational complexity of the proposed method. Finally, we demonstrate the superiority of quadratic projective regression against Factorization Machines by experiment on real data.

Keywords and phrases: supervised learning; stochastic optimization; high dimensions; quadratic regression; projective methods.

Computing Classification System 1998: G.1.6, G.1.2

Mathematics Subject Classification 2010: 68T05, 68W25

References

1. Smola, A., Vishwanathan, S. V. N. (2008), *Introduction to machine learning*, Cambridge University Press, UK, 226 p.

2. Witten, I. H., Frank, E. (2005), *Data Mining: Practical machine learning tools and techniques, 3rd edition*, Morgan Kaufmann, Waltham, US, 664 p.

3. Vapnik, V. N., Chervonekis, A. Y. (1974), *Pattern recognition theory [Teoria raspoznavania obrazov]*, Nauka, M., 416 p.

4. Friedman, J., Hastie, T., Tibshirani, R. (2009), *The Elements of Statistical Learning, 2nd edition*, Springer, Berlin, 745 p.

5. Harris, D., Harris, S. (2012), *Digital design and computer architecture, 2nd edition*, Morgan Kaufmann, Waltham, US, 712 p.

6. Rendle, S. (2010), “Factorization machines”, *Proc. 10th IEEE Conf. On Data Mining (ICDM) (Sydney, Australia)*, pp. 995–1000.

7. Rendle, S. (2012), “Factorization machines with libFM”, *ACM Transactions on Intelligent Systems and Technology (TIST)*, Vol. 3, No. 3, pp. 57–79.

8. Granichin, O. N., Polyak, B. T. (2003), *Randomized estimation and optimization algorithms under almost arbitrary noises [Randomizironannie algoritmi otsenivania i optimizatsii pri pochti proizvolnih pomehah]*, Nauka, M., 290 p.

9. Friedman, J. H. (1997), “On bias, variance, 0/1–loss, and the curse-of-dimensionality”, *Data mining and knowledge discovery*, Vol. 1, No. 1, pp. 55–77.

10. Friedman, J. H., Stuetzle, W. (1981), “Projection pursuit regression”, *Journal of the American statistical Association*, Vol. 76, No. 376, pp. 817–823.

11. Granichin, O. N. (2003), *Introduction in stochastic optimization and estimation methods. Schoolbook [Vvedenie v metodi stohasticheckoi optimizatsii i otsenivania. Uchebnoe posobie]*, S.-Petersburg State Univ. Publ., 131 p.

12. Bottou, L. (1981), “Large-scale machine learning with stochastic gradient descent”, *Proc. COMPSTAT (Paris, France)*, pp. 177–186.

13. GroupLens Research (2015), *European Football Championship 2012*, available at: http://grouplens.org/datasets/movielens/

14. Van Rossum, G., Drake, F. L. (2015), *Python Reference Manual*, available at: http://www.python.org

15. Jones, E., Oliphant, T., Peterson, P. (2015), {*SciPy*}*: Open source scientific tools for* {*Python*}, available at: http://www.scipy.org

СВЕДЕНИЯ ОБ АВТОРАХ

Михаил Александрович БЕЛЯЕВ – аспирант кафедры Информатики и вычислительной математики Механико-математического факультета Самарского государственного аэрокосмического университета им. академика С. П. Королёва.

Елена Игоревна БОЛЬШАКОВА – кандидат физико-математических наук, доцент кафедры Алгоритмических языков факультета Вычислительной математики и кибернетики Московского государственного университета им. М. В. Ломоносова.

Виталий Ильич ЛЕВИН – доктор технических наук, профессор Пензенского государственного технологического университета.

Василий Иванович ЛОСЕВ – независимый исследователь.

Борис Феликсович МЕЛЬНИКОВ – доктор физико-математических наук, профессор, заведующий кафедрой Прикладной математики и информатики Тольяттинского филиала Самарского государственного аэрокосмического университета им. академика С. П. Королёва.

Елена Анатольевна МЕЛЬНИКОВА – кандидат физико-математических наук, доцент кафедры Прикладной математики и информатики Тольяттинского филиала Самарского государственного аэрокосмического университета им. академика С. П. Королёва.

Алексей Анатольевич НОСКОВ – аспирант кафедры Алгоритмических языков факультета Вычислительной математики и кибернетики Московского государственного университета им. М. В. Ломоносова.

Александр Алексеевич СЕНОВ – аспирант кафедры Системного программирования Математико-механического факультета Санкт-Петербургского государственного университета.

ABOUT THE AUTHORS

Mikhail A. BELYAEV – post-graduate student of Department “Informatics and Computing Mathematics”, Faculty of Mechanics and Mathematics, Samara State Aerospace University (Russia).

Elena I. BOLSHAKOVA – candidate of physical and mathematical sciences, associate professor of Department “Algorithmic Languages”, Faculty of Computational Mathematics and Cybernetics, Moscow State Lomonosov University (Russia).

Vitaly I. LEVIN – doctor of technical sciences, professor of Penza State Technological University (Russia).

Vasily I. LOSEV – independent researcher (Russia).

Boris F. MELNIKOV – doctor of physical and mathematical sciences, professor, head of Department “Applied Mathematics and Informatics”, Togliatti branch of Samara State Aerospace University (Russia).

Elena A. MELNIKOVA – candidate of physical and mathematical sciences, associate professor of Department “Applied Mathematics and Informatics”, Togliatti branch of Samara State Aerospace University (Russia).

Alexey A. NOSKOV – post-graduate student of Department “Algorithmic Languages”, Faculty of Computational Mathematics and Cybernetics, Moscow State Lomonosov University (Russia).

Alexander A. SENOV – post-graduate student of Department “Software Engineering”, Faculty of Mathematics and Mechanics, Saint-Petersburg State University (Russia).